

FOURTH SEMIANNUAL TECHNICAL REPORT

(15 June 1971 - 15 December 1971)

AD737403

FOR THE PROJECT

"RESEARCH IN

STORE AND FORWARD COMPUTER NETWORKS"

Principal Investigator

and Project Manager:

HOWARD FRANK (516) 671-9580

ARPA Order No. 1523

Contractor: Network Analysis Corporation

Contract No. DAHC 15-70-C-0120

Effective Date: 15 October 1969

Expiration Date: 15 October 1972

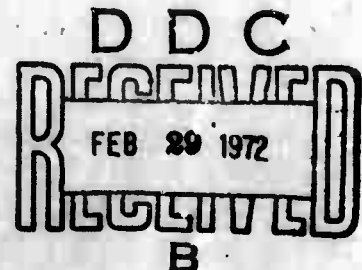
DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

Sponsored by

Advanced Research Projects Agency

Department of Defense



The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

128

**BEST
AVAILABLE COPY**

SUMMARY

Technical Problem

The Network Analysis Corporation contract with the Advanced Research Projects Agency incorporates the following objectives: To determine the most economical configurations for the ARPA Computer Network; to study the properties of store-and-forward networks, and in particular to investigate the relationship between traffic, routing, throughput, and cost for large networks; and finally, to develop procedures for the analysis and design of reliable and survivable computer and communication networks.

General Methodology

The study of ARPA Net design properties has heavily used the NAC computer network design programs for generating low cost systems. The approach to the reliability problem has combined analysis, combinatorics and computer simulation to derive efficient reliability analysis schemes. The heart of the research program has been a dual attack on basic network theoretical problems and the development of computational techniques for efficiently handling large network structures.

Technical Results

Some of the results accomplished during the reporting period are:

- A projected 26 node design is shown to be within 1% of a theoretical globally optimum solution and a projected 40 node design using hypothetical node locations indicates that the economic trends exhibited by the previous evolution of the ARPA Net can be expected to continue.
- A detailed throughput reliability analysis of the ARPA Net considering element failures, traffic requirements, routing, and acceptable delays shows that the Network is highly adaptable to component failures.
- It is shown that the high peak bandwidths presently achievable within the Network are obtained at virtually no cost.
- A new routing procedure yielding throughputs extremely close to the optimal during heavy traffic is described. This new algorithm represents a major computational breakthrough for the "shortest path problem," which is one of the most fundamental of network analysis.
- Major computational improvements for large network reliability analysis are described.

Department of Defense Implications

The new technical results extend our previous conclusions about the economic viability and practicality of ARPA-like networks for DOD communications. The results on reliability increase the range of networks that can be studied to include very large networks of the kind that would be required for Defense Communications. The new computational results on routing, shortest paths, and connectivity will increase the savings resulting from optimization of these networks.

Implications for Further Research

The present results point to the need and aid in the progress of research in a number of areas including: determining the relationship between network connectivity, size, reliability and survivability; developing new optimization algorithms for very large network design; developing design procedures for large networks which specifically incorporate reliability/availability requirements as constraints. Furthermore, the significant computational improvements in the connectivity and shortest path algorithms open new possibilities for the improved optimization of other network structures such as large scale systems of Telpaks and other leased line options.

TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
1. DESIGN PROBLEMS FOR COMPUTER NETWORKS.....	1
2. ARPA NETWORK ANALYSIS AND DESIGN	
2.1 ARPA NETWORK GROWTH.....	12
2.2 ARPA NETWORK THROUGHPUT RELIABILITY ANALYSIS.....	25
2.3 ECONOMIC CONSIDERATIONS.....	39
3. ROUTING STRATEGIES FOR COMPUTER NETWORK DESIGN	
3.1 THE ROUTING PROBLEM.....	58
3.2 A MODIFIED ROUTING ALGORITHM FOR ARPA-LIKE NETWORKS.....	61
3.3 NEW SHORTEST ROUTE ALGORITHMS.....	77
3.4 COMPUTATIONAL COMPARISONS.....	84
4. RELIABILITY OF LARGE COMPUTER NETWORKS	
4.1 INTRODUCTION.....	96
4.2 NETWORK RELIABILITY AS A FUNCTION OF SIZE.....	97
4.3 COMPUTATIONAL CONSEQUENCES OF NETWORK DECOMPOSITION.....	100
4.4 FAST ALGORITHMS FOR COMPUTING COMPONENTS OF SPARSE NETWORKS.....	105

1.

DESIGN PROBLEMS FOR COMPUTER NETWORKS

The present Network Analysis Corporation contract with the Advanced Research Projects Agency incorporates the following objectives:

- To determine the most economical configurations for the ARPA Computer Network.
- To study the properties of store-and-forward networks, and in particular to investigate the relationship between traffic, routing, throughput, and cost.
- To develop procedures for the analysis and design of reliable and survivable computer and communication networks.

The research effort has resulted in a constantly evolving network optimization computer program which is able to produce extremely economical networks. The program's capabilities have been advanced to the point where networks with several hundred nodes can be handled. Cost-throughput characteristics for a 200-node store-and-forward network were determined. These characteristics extend the results of previous studies which showed that large ARPA-like networks are economical to operate using the present equipment of the ARPA Net.

For the ARPA Computer Network, low cost networks have been derived and augmented as the network has grown. It has been shown that the ARPA network provides near optimal performance and retains

its high throughput capabilities under variations in input traffic rates.

Activity in the reliability and survivability areas has focused on formulating realistic network survivability criteria and developing procedures for analyzing and designing large networks. The effort has resulted in the development of analysis methods more than 1000 times more efficient than conventional schemes.

Many formidable problems remain in the optimization of the design of computer networks. These problems which are under study at NAC include:

Network Choice. In general, there are $\frac{N(N-1)}{2}! \left[\frac{N(N-1)}{2} - M \right]! / M!$ ways of arranging M links among N nodes. Considering all possible designs by computer is out of the question, and no known theoretical method exists for finding an optimal computer-communication layout.

Discrete Elements. Components usually are available in discrete sizes. Thus, line speeds can be at 2000, 2400, 3600, ..., 9600, ..., 50,000, 240,000 baud, etc. This means an integer optimization problem must be solved. Except in special cases, no theoretical methods now exist for problems of practical size.

Nonlinearities. Component cost structures, time delays, and reliabilities are all nonlinear functions. Typical cost functions are neither "concave" nor "convex" and no analytical methods are

available to obtain optimal solutions for networks containing such elements.

Present methods for topological optimization involve the heuristic application of a family of optimization procedures called "branch exchanges." Branch exchange methods are search procedures and for network designs incorporating realistic constraints on routing, throughput, delay, cost, reliability and physical implementation, the complexity of the computation is on the order of n^3 to n^6 where n is the number of nodes. For networks with several hundred nodes, present procedures are inadequate because of the large amounts of computer time needed to perform the optimization. Therefore, entirely new classes of optimization techniques must be developed for large scale networks. Research on these new procedures is now underway.

Routing. For a centralized network, the routing problem reduces to a flow control problem since there is only one path between any pair of nodes. On the other hand, for distributed networks like the ARPA network, two different kinds of routing problems must be solved to specify and operate an efficient network. Routing procedures used for design must be effective for finding low cost networks. However, once a particular network structure has been selected, it becomes essential to take full advantage of the location and capacity

of each line and node. Furthermore, the routing procedures installed in the physical network must (1) keep network "overhead" low and (2) adjust to variations in traffic as a function of time. For the design problem traffic flows are usually assumed to be time invariant. A variety of alternate routing procedures which can be rapidly performed by computer are then possible. These procedures must model the dynamic routing strategies which are actually implemented in the network. Furthermore, the performance of a particular routing strategy is critically dependent on the capacities assigned to the links. In many procedures either these capacities are assigned before the routing is performed, or a routing is first performed and capacities then assigned. The difficulty with such approaches is that a priori the routing-capacity assignment combination is biased and often any changes of effecting economics are reduced. Thus, problems of importance are: (1) to find routing procedures which are realistic and simultaneously can be run in milliseconds during optimization routines, and (2) that no general procedures are yet known for assigning capacities from a wide range of options while simultaneously optimizing the routing used in the design procedure. We view this as one of the most important open problems.

Clustering. The design of small networks (say, less than 100 nodes) is relatively straightforward. However, the design for larger networks is under intensive study. At present, it appears

that the most fruitful approach to design and implementation will be based on the partitioning of the network into regions, or equivalently, constructing a large network by connecting a number of regional networks. To send a message, a sender might specify (1) the destination region and (2) the destination node in that region. In order to create effective partitions, we must solve a variety of "clustering" problems.

Nodes may be clustered into regions for numerous reasons such as (a) to partition status information for use in routing, flow control, and other decision process within the operating network; (b) to determine regions of low, medium and high speed lines in hierarchical structures; (c) to determine decompositions for topological designs, and (d) to find concentrator/multiplexer locations. The literature on clustering and partitioning is large but fragmented and spread over many domains including information retrieval, taxonomy and networks. A potentially valuable research area is the application of known clustering techniques to computer networks. This requires the assignment of appropriate "distance measures" to take into account cost, capacity, traffic, delay, reliability and routing. Almost no theoretical results are presently known for this problem.

Reliability. An essential characteristic of any good network design is that it not suffer significant degradation in performance if some elements fail. With the state of current technology, both nodes and communication links have nontrivial downtimes. Therefore, the network design must provide for these failures by having sufficient alternate paths to satisfy flow requirements and time delay constraints. For example, some networks are made reliable by installing parallel nodes and lines while the ARPA design utilizes distributed control and multiple independent paths between each pair of nodes.

The network reliability problem has two aspects--analysis and design. Reliability analysis, even for large nets, has now been advanced to the point where it appears relatively straightforward. Preliminary studies indicate that for large networks, reliability may be the dominant design constraint. However, the development of design techniques which handle realistic reliability constraints is still in its infancy.

In this report, we summarize NAC's studies of computer network analysis and design during the period 15 June 1971 to 15 December 1971.

ARPA Net Studies

Chapter 2 describes NAC's studies of the economic and growth characteristics of the ARPA Network. In Section 2.1, the continued evolution of the Network is discussed and it is seen that a projected 26 node design is within 1% of the theoretically globally optimum but unrealizable solution that would exist if all nodes were to be connected at the same time without the constraint of an installation schedule. Of equal importance, a projected 40 node design using hypothetical node locations indicates that the economic trends exhibited by the previous evolution of the Net can be expected to continue.

Section 2.2 provides a detailed throughput reliability analysis of the ARPA Net. This analysis considers element failures, traffic requirements, routing, and acceptable delays as well as other pertinent network characteristics. This study shows that as long as a pair of nodes can sustain any communication, the network will have sufficient capacity and routing algorithms will be able to utilize this capacity to attain throughputs near the ideal design levels achievable under failure free conditions.

Section 2.3 considers two problems: (1) The cost of providing the peak bandwidths of 85 KBPS per node pair that are presently realizable in the ARPA Net; and (2) The incremental costs for adding

"user type" nodes to the ARPA Net. For the first problem, it is shown that only insignificant savings can be achieved by allowing major reductions in the ARPA Net's peak bandwidth capabilities. For the second problem, cost-performance tradeoffs for adding new network nodes are established.

Routing

The objective that time delay be minimized subject to a set of flow constraints makes the routing problem a variation of a nonlinear multicommodity flow problem. This problem, which was discussed in our Second Semiannual Report, can be readily formulated as a separable convex programming problem with the delay as the objective function and the conservation of flow and capacity limitations as the constraints; but for networks with more than a few nodes it is not computationally efficient to solve the programming problem, and this approach cannot be used during the design stage.

A heuristic routing procedure (described in Semiannual Reports 1 and 2) routes flow over the least utilized paths containing a minimum number of nodes. This approach yields throughputs within 5%-20% of optimum; and in addition to being fast (over three orders of magnitude faster than the programming approach), it facilitates minor changes of the network structure. On the other hand, it does not take good advantage of possible split routing

and variations in link capacities, and leaves room for considerable improvements especially in the case when the network contains a wide distribution of different line capacities. (An apparently desirable characteristic of very large networks.)

A generalization of the above routing procedure is discussed in detail in Section 3.2 of Chapter 3. Different types of flows are simultaneously routed over the paths with minimum numbers of nodes. (These paths are "shortest paths" using a simple unit metric for each line.) When no shortest path with excess capacity is available, the saturated lines are deleted from the network and flows are then routed over the shortest paths of the remaining subnetwork. The process is continued until the network is disconnected. The new procedure yields throughputs extremely close to the optimal during heavy traffic. Furthermore, the routing strategy is very similar to physical routing schemes under study for the ARPA Network, and it exhibits analogous behavioral properties.

Equally important, the new routing algorithm represents a major computational breakthrough for the "shortest path problem." The problem is one of the most fundamental ones of network analysis. New algorithms for this problem, which appear to be the most efficient yet devised, are given in Section 3.3. In Section 3.4

the computational efficiency of the new algorithms are compared with the previously accepted "best" algorithm.

Reliability

Due to the success and economic potential of the "ARPA-like" networks, the size of computer networks can be expected to increase rapidly. A large number of computers in a computer network gives rise to several questions about its reliability. For the ARPA Net, the principal reliability requirement for network designs of less than 30 or 40 nodes is that there exist two node disjoint paths between every pair of nodes. This guarantees that at least two nodes or links must fail before any two nodes cannot communicate with one another. Many detailed reliability analyses of networks designed in this way indicated that this approach guarantees sufficient reliability for the network taken as a whole for nets similar to the current ARPA Net. The first question of interest is: does the two node disjoint path method suffice as the number of nodes grows? Closely related to this question is: what minimum amount of network investment is required, as the number of nodes increases, in order to maintain a given level of network reliability. In Section 4.2 preliminary results bearing on these questions are given. Simply stated, it does not appear that network reliability constraints can be met for very large nets simply by requiring two node disjoint paths between each pair. More sophisticated techniques for

designing large nets will be required. Such techniques are currently under investigation.

For many reasons it seems imperative that large computer nets will exhibit some hierarchical or decomposed structure. In Section 4.3 the computational consequences for reliability analysis of a two level hierarchical approach is investigated in which the nodes are partitioned into subnetworks called "regions" interconnected by a "global" network. In the same section, the related question of what size regions yields minimum computation for analysis is explored.

Finally, to make reliability analysis of large networks feasible, computational improvements over the techniques employed for the smaller networks must be developed. A detailed analysis of the growth of computation with network size for various reliability analysis techniques is found in Section 4.4, and new techniques that are considerably faster than previously known techniques are specified.

During the reporting period, a number of optimizations were performed to introduce new nodes into the ARPA net, to test and improve overall network economy and reliability, and to study the economic and growth characteristics of the ARPA network. The results of these studies are summarized in the following sections.

2.1 ARPA NETWORK GROWTH

Since initially there was no clear knowledge of the total traffic the network would have to accommodate, the network was first constructed with enough capacity to accommodate any reasonable traffic requirements. At the initial stages of the design, the "two-connected" reliability constraint forced the network throughput to be in the range 10-15 KBPS/node since two communication paths between every pair of IMPs is needed. As new IMPs are added to the network, the capacity is being systematically reduced until the traffic occupies a substantial fraction of the network's total capacity. At this point, the network's capacity will be increased to maintain a desired percentage of loading. To insure that this process can be efficiently performed, each basic configuration is designed so that additional links can be added to economically increase network throughput.

If the locations of all network nodes are known in advance, it is clearly most efficient to design the topological structure as a single global effort. However, in the ARPA Net, as in most actual networks, node locations are added and modified on numerous occasions. On each such occasion, the topology could be completely reoptimized to determine a new set of link locations.

In practice, however, there is a long lead time between the ordering and the delivery of a link and major topological modifications cannot be made without substantial difficulty. It is therefore prudent to add or delete nodes with as little disturbance as possible to the basic network structure consistent with overall economic operation. Figure 2.1.2(a) shows the 26-node ARPA Net derived using the policy of adding new nodes with minimum disturbance to the basic 15-node configuration existing in March 1971 in the least costly manner. This 26-node net has a link cost of approximately \$880,000 per year, a throughput of about 9.9 KBPS/node, and an average availability of communication paths between 96% of all node pairs.

At approximately 26 nodes the growth pattern within the net makes it desirable to implement some fundamental changes in network structure. The original net expanded eastward from a 4-node configuration on the West Coast. Because of this origination, the West Coast had somewhat more capacity than other parts of the

country. Also, because of the excellent relative location of the UTAH node, two of the three planned cross country paths utilize this node thus creating a great dependence in the enlarged net. Finally, the expanded net has a number of new nodes in the Washington, D.C. area. A redesign of the network taking advantage of these facts is shown in Figure 4.2.2(b). The new network design was aimed at changing as few existing or ordered links as possible, maintaining a throughput of around 10 KBPS/node, and increasing network reliability.

The new network design, which is redrawn for visual convenience in Figure 2.2.2(c) has a link cost of \$810,000 per year (\$70,000 per year lower). Significantly, the network reliability has also been increased to a point where further substantial increases can be made only by either adding many new links or reducing IMP downtimes. A graph of reliability versus element downtime is shown in Figure 2.1.2(d) for the designs in (a) and (b).

To test the overall economy of the design shown in Figures 4.2.2(b) and (c), another design was produced. The additional design was generated under the assumption that all 26 nodes were to be interconnected into a network at the same time, with no restrictions on link locations. This design, which represents a "global" optimization, is shown in Figure 4.2.2(e). This topology, which is believed optimal under uniform traffic requirements, has link costs of \$800,000

per year. Thus, the actual projected 26-node ARPA Net design is within \$10,000 dollars (less than \$400/node/year) of a theoretically globally optimum but unrealizable solution.

Figures 4.2.2(f) and (g) show respectively 30 and 40 node net designs using proposed and hypothetical node locations. (See Table 2.2.1). These designs show that the performance and economic trends indicated by the growth of the net from ten nodes to the presently planned net can be expected to continue. The relationship between cost to number of nodes is shown in Figure 2.2.1. During the growth of the net, the link cost per node has been reduced from \$44,000/node/year for the 15-node net to \$30,000/node/year for the 26-node net while the design throughput level has been reduced only from 12.7 KBPS/node to 9.9 KBPS/node for the same nets.

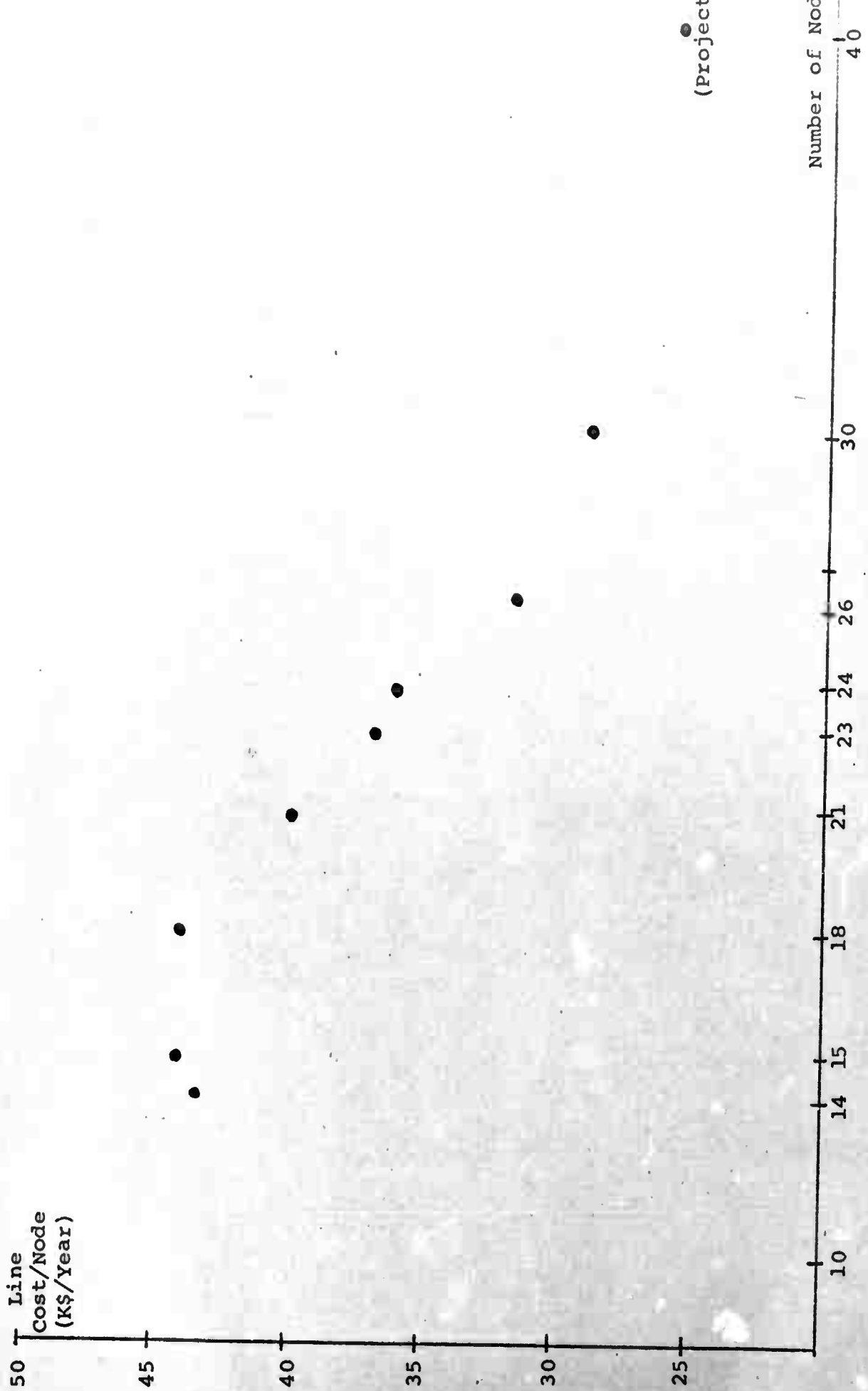


FIGURE 2.1.1.1

EVOLUTION OF LINE COST PER NODE FOR ARPA NET

TABLE 2.1.1

NODE COORDINATES
OPERATIONAL OR PROPOSED LOCATIONS

Node Number	Node Name	Node Location	
		Latitude	Longitude
1	UCLA	34 04	118 31
2	SRI	37 22	122 10
3	UCSB	34 30	119 45
4	UTAH	40 40	111 50
5	RAND	34 00	118 35
6	BBN	42 30	71 20
7	SDC	34 01	118 33
8	MAC	42 30	71 12
9	ILLINOIS	40 05	88 30
10	HARVARD	42 30	71 15
11	CARNEGIE-MELLON	40 30	79 50
12	ETAC (WASHINGTON)	38 50	77 00
13	SAAC	38 55	77 10
14	LINCOLN LABS	42 35	71 20
15	CASE	41 30	81 45
16	STANFORD	37 18	122 10
17	MITRE	39 00	77 00
18	NCAR DENVER	39 30	105 00
19	UCD	38 39	121 45
20	AFWS OMAHA	41 00	96 00
21	ROME, N.Y.	43 15	75 25
22	NASA	37 17	122 02
23	USC	34 00	118 21
24	TINKER	35 27	97 32
25	McCLELLAN	38 35	121 30
26	NBS	39 08	77 10
27	NEW YORK	41 15	73 47
28	UCSD	32 55	117 20
29	ABERDEEN	38 60	77 0
30	FT. BELVOIR	38 65	77 0

Rank Among
Largest
Cities

HYPOTHETICAL LOCATIONS

31	3	CHICAGO	47 49	87 37
32	4	PHILADELPHIA	40 0	75 13
33	5	DETROIT	42 22	83 10
34	9	ST. LOUIS	38 39	90 15
35	14	MINNEAPOLIS	44 58	93 15
36	15	BUFFALO	42 54	78 71
37	16	HOUSTON	29 46	95 21
38	17	MILWAUKEE	43 10	87 56
39	19	SEATTLE	47 36	122 20
40	20	DALLAS	32 45	96 48

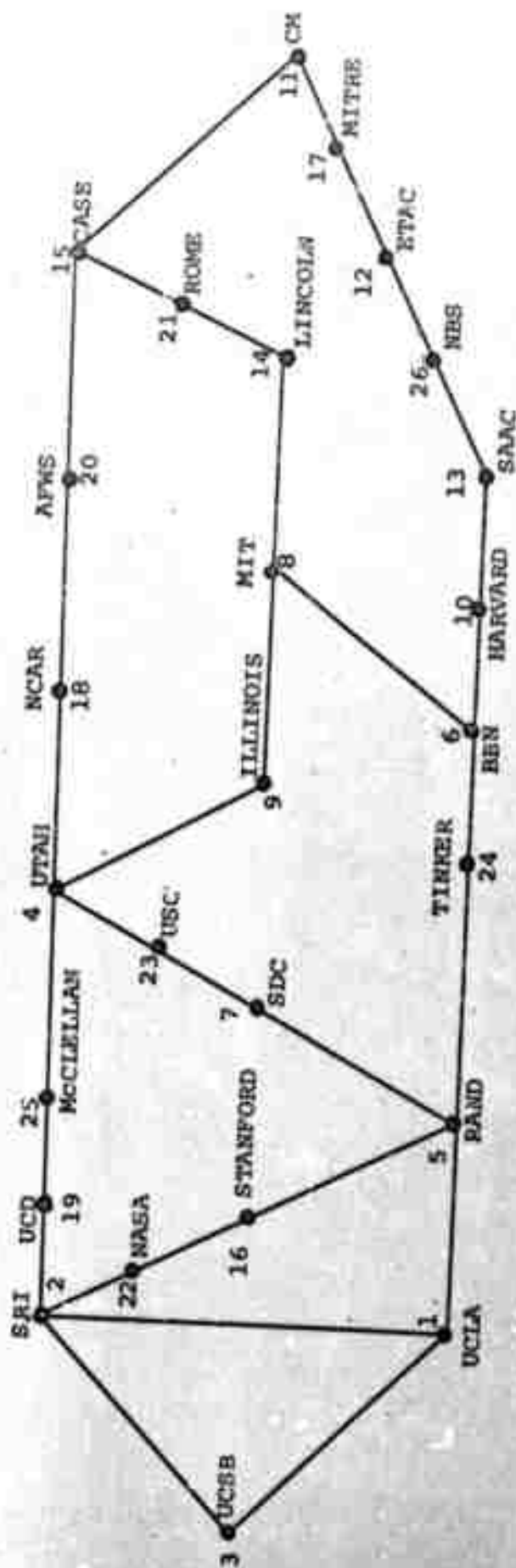


FIGURE 2.1.2(a)

26-Node Net Derived Without Modifying Basic Structure

Cost: \$880,000/year

Throughput: 9.9 KBPS/node
(Uniform traffic)

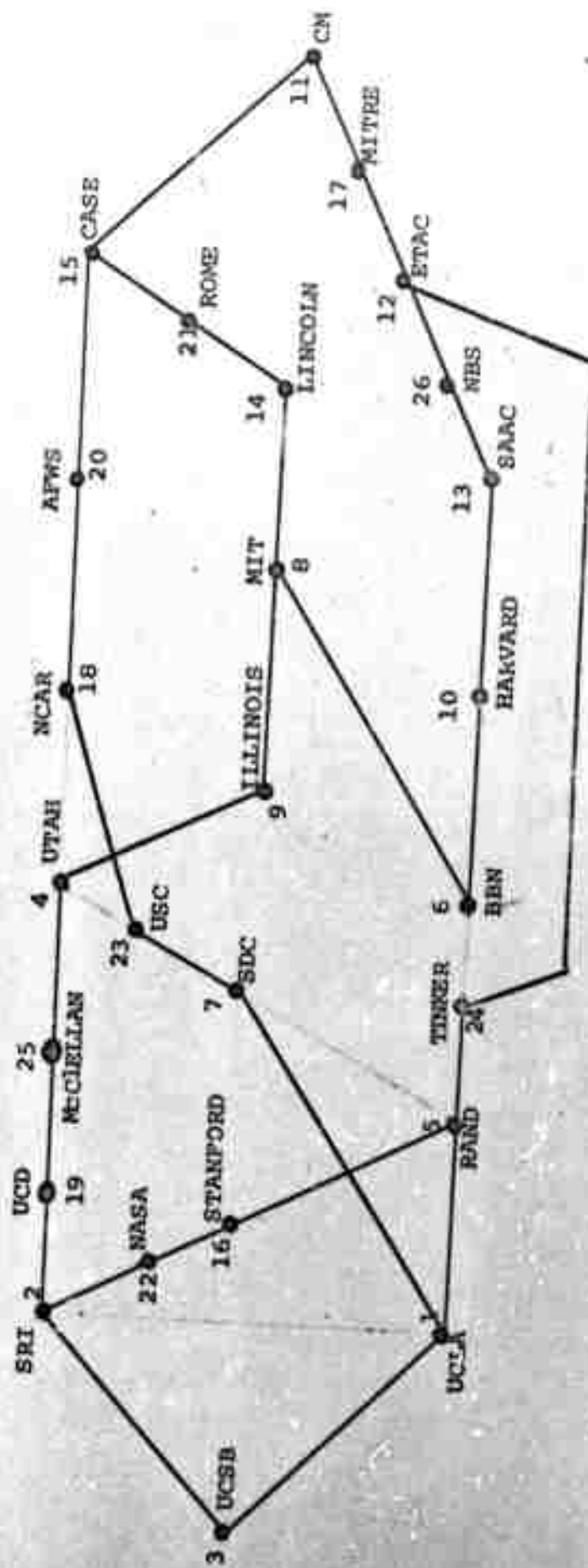


FIGURE 2.1.2(b)

26-Node Not Derived by Slightly Modifying Basic Structure

Cost: \$810,000/year
Throughput: 9.9 KBPS/node
(Uniform traffic)

Fraction of Node
Pairs Disconnected

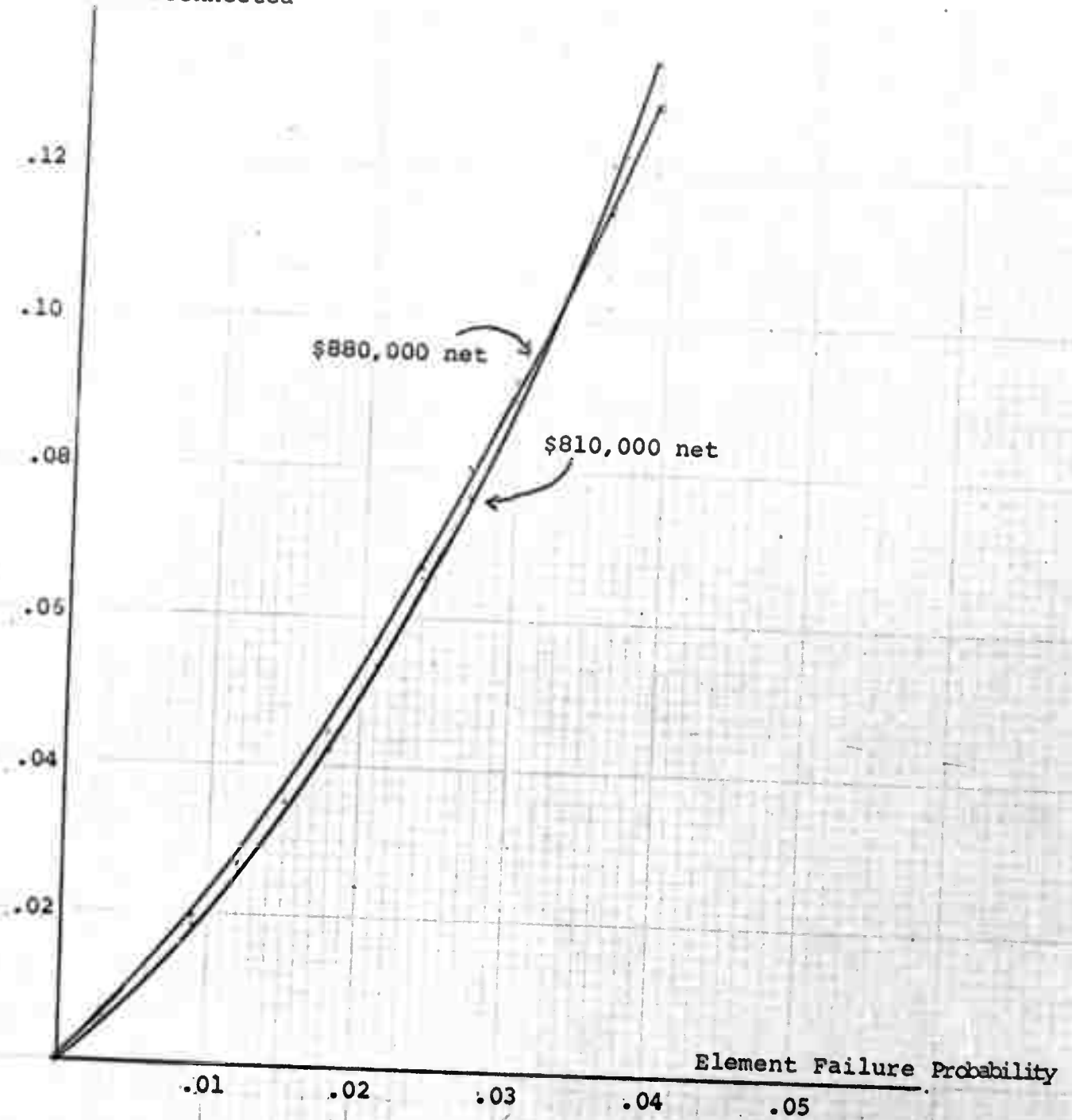


FIGURE 2.1.2(d)

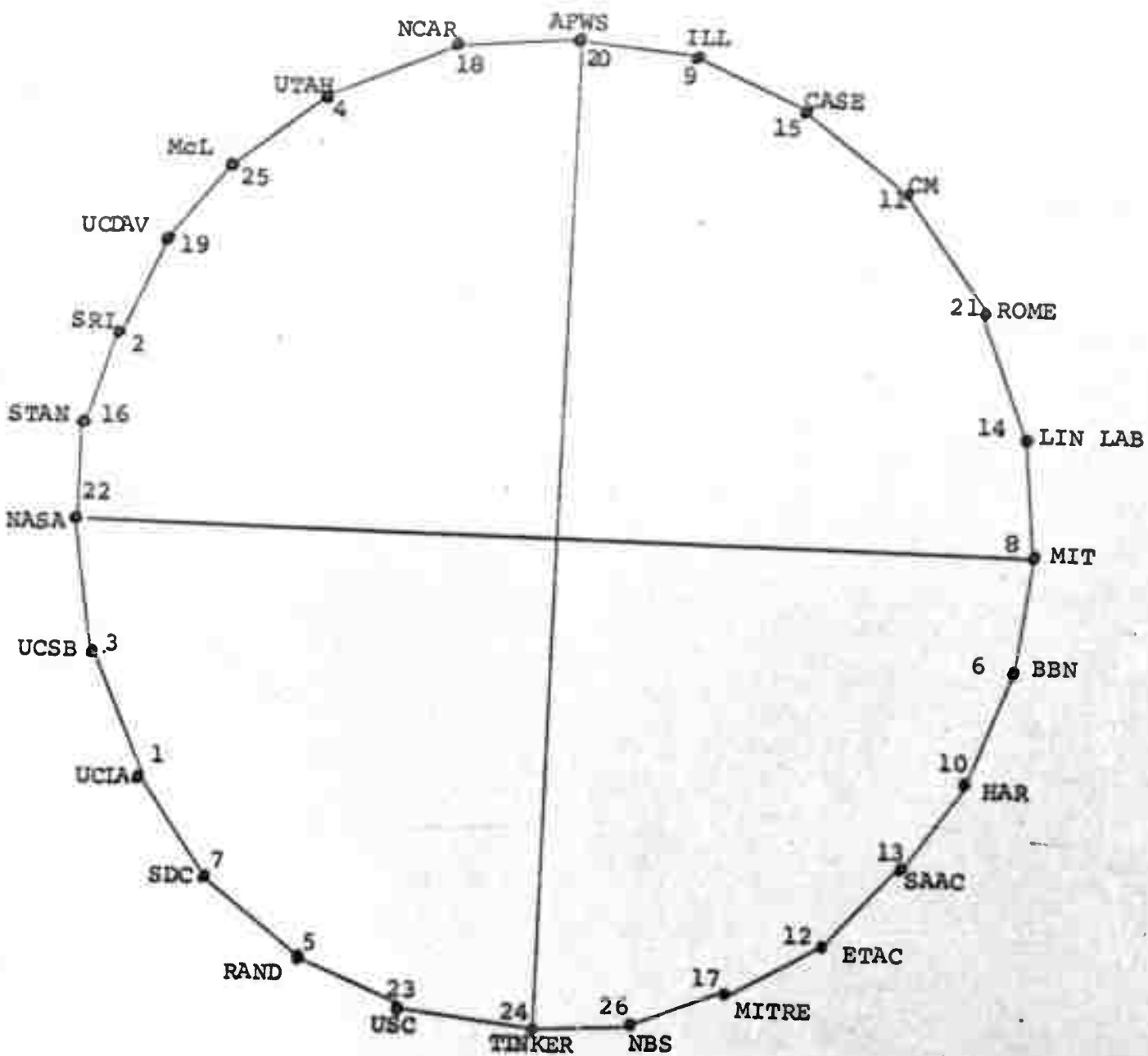


FIGURE 2.1.2(e)

In Unconstrained 26-Node Net Optimization

Cost: \$800,000/year

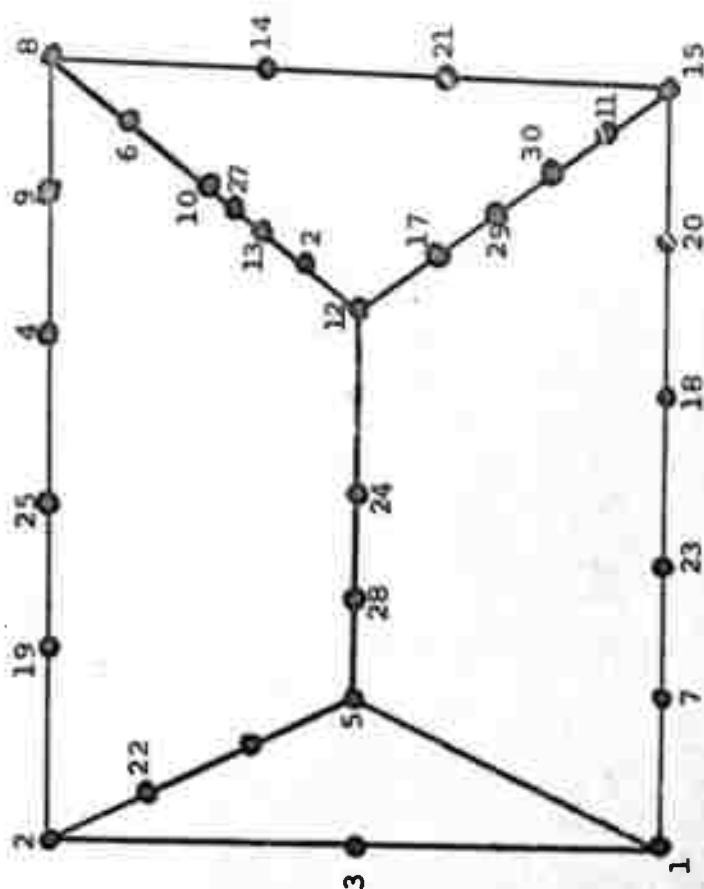


FIGURE 2.2.2(f)

A Projected 30 Node Net

Cost:	\$859,000/year
Throughput: (uniform traffic)	8.7 KBPS/node (50 K Packets/hr/node)
Node Pair Availability:	94.2%

Projected 40-Node Topology
(With hypothetical Node Locations)

Cost: \$1,025,000/year
Throughput: 6.0 KBPS/node (34 Kpackets/hr/node)
Node Pair Availability: 92.5%

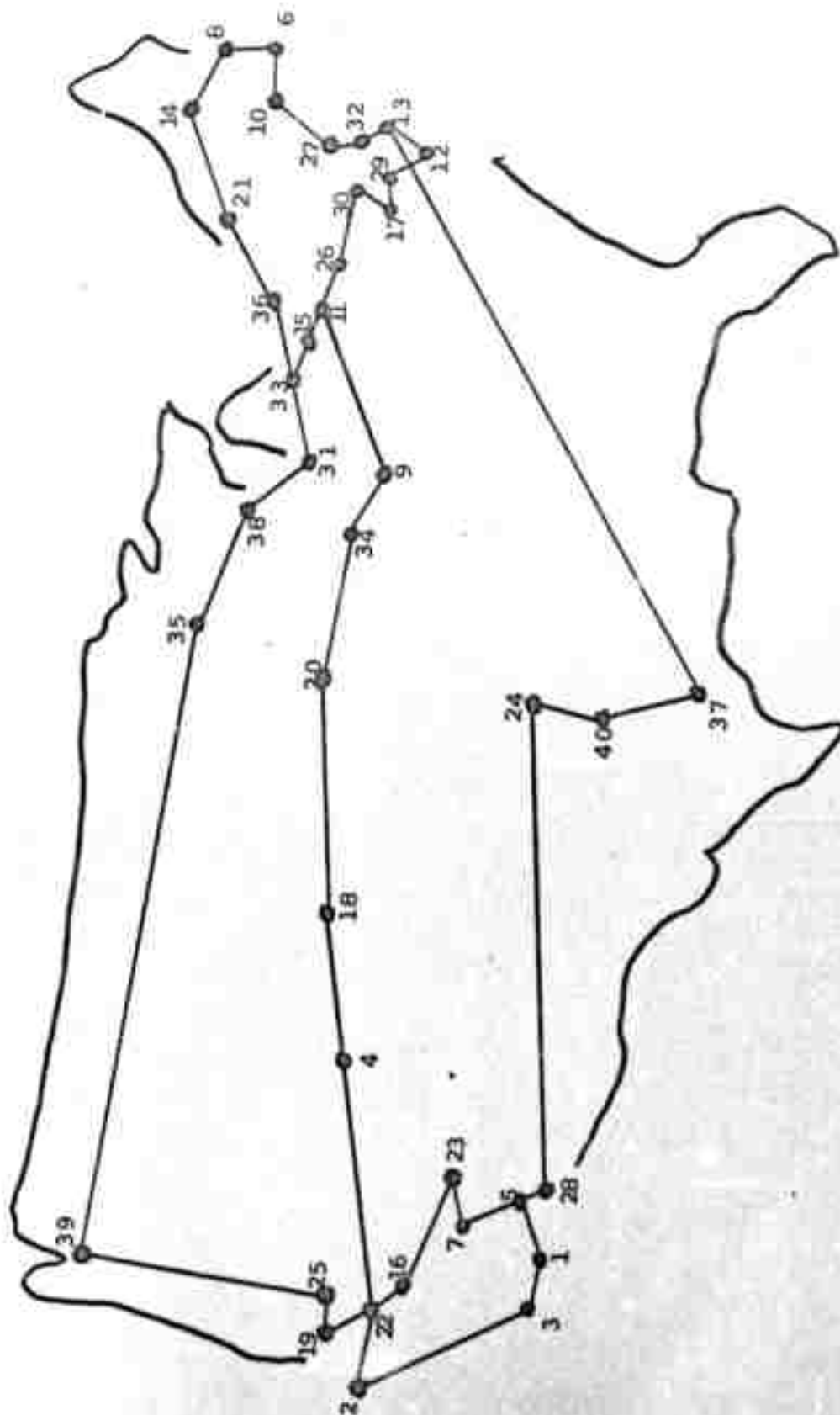


FIGURE 2.1.2(g)

2.2 ARPA NET THROUGHPUT RELIABILITY ANALYSIS

A simple and natural characterization of network reliability is the ability of the network to sustain communication between all operable pairs of nodes. For design purposes, the requirement of two independent paths between nodes insures that at least two nodes and/or links must fail before any pair of operable IMPs cannot communicate. This criterion is independent of the properties of the nodes and links, and does not take into account the "degree" of disruption that may occur. Hence, it does not reflect the actual availability of resources in the net.

A more refined measure is the average fraction of node pairs that cannot communicate because of node and link failures. To calculate this measure, knowledge of the element failure rates must be available or estimated. If desired, the availability of specific nodes and the existence of specified communicating node pairs can also be considered using this formulation. However, the "expected fraction of noncommunicating node pairs" is still a purely topological reliability measure which does not completely reflect the "adequacy" of a network operating without some of its components.

The most detailed level of analysis of reliability incorporates element failures, flow requirements, routing, acceptable delays

and other pertinent network characteristics. In order to test the adequacy of the ARPA Net under the most stringent of conditions, a reliability analysis treating these factors was performed.

The 23-node ARPA Net studied in detail in the Third Semi-annual Report was considered to maintain continuity of treatment. The effect on throughput at average delay 0.2 seconds was examined by removing nodes and links from the network and applying the NAC routing and analysis algorithms to the remaining network. To perform a total analysis, an unmanagable number of computations would be required. Therefore, throughputs for only one and two node failures and one and two link failures were calculated. To be conservative, it was assumed that all removals of three components or more would reduce throughput to zero. That is, any combination of three or more elements was assumed to be a cut. Since there are many such combinations which are not cuts (for example, only about 25% of the possible three link combinations are cuts), the actual average network throughput is slightly higher than the numbers given below.

Table 2.2.1 and 2.2.2 show the effects of link and node failures on network throughput under the assumption that all traffic requirements are equal. The nominal throughput of the 23 node net with all elements operable is 11.5 KBPS/node.

Table 2.2.1 shows the effect for link failures while Table 2.2.2 shows the effect for node failures for element downtimes of 2% (The estimates for downtimes in the operating network). Table 2.2.3 and 2.2.4 itemize the basic data summarized in the first two tables. Combining both the node and link failure effects yields an average throughput of approximately 9.0 KBPS/node. Thus, the traffic handling capability of the operating net can be expected to be close to the nominal throughput under ideal conditions. (In fact, the results of flow sensitivity analyses under varying traffic requirements and perfect component operations indicate throughput variations of 10%-20% are to be expected under even ideal conditions.) An alternative interpretation is that as long as a pair of nodes is able to communicate at all through the net, the network will have (1) sufficient capacity and (2) the routing algorithms will be able to utilize this capacity to provide acceptable throughputs.

TABLE 2.2.1

EFFECTS OF LINK FAILURES ON THROUGHPUT

Link Failure Probability = 0.02

Expected Throughput = 10.1 KBPS/node

I. One link failed

Sample mean = 9.1 KBPS/node Sample deviation = 1.45 KBPS/node

Empirical distribution function:

Throughput less than	:	6	7	8	9	10	11	12
%	:	0	7.1	25.0	46.4	67.9	85.7	100

Maximum throughput = 11.5 KBPS/node Minimum throughput = 6.8 KBPS/node

II. Two links failed

Sample mean = 7.0 KBPS/node Sample deviation = 2.57 KBPS/node

Empirical distribution function:

Throughput less than	:	0	1	2	3	4	5	6	7	8
%	:	0	7.9	7.9	7.9	7.9	18.2	25.9	41.2	60.3
		9	10	11	12					
		83.1	94.6	99.5	100					

Maximum throughput = 11.2 KBPS/node Minimum throughput = 0:

TABLE 2.2.2

EFFECTS OF NODE FAILURES ON THROUGHPUT

Node Failure Probability = 0.02

Expected throughput = 9.9 KBPS/node

I. One node failed

Sample mean = 8.5 KBPS/node Sample deviation = 1.64 KBPS/node

Empirical distribution function:

Throughput less than	:	4	5	6	7	8	9	10	11	12
%	:	0	4.3	4.3	17.4	39.1	56.5	82.6	91.3	100

Maximum throughput = 11.10 KBPS/node Minimum throughput = 6.27 KBPS/node

II. Two nodes failed

Sample mean = 6.1 KBPS/node Sample deviation = 2.72 KBPS/node

Empirical distribution function:

Throughput less than	:	0	1	2	3	4	5	6	7	8
%	:	0	12.6	12.6	12.6	12.6	33.6	38.3	51.0	71.9
		9	10	11						
		90.1	97.6	100						

Maximum throughput = 10.80 KBPS/node Minimum throughput = 0.

TABLE 2.2.3

EFFECTS OF LINK FAILURES ON THROUGHPUT

<u>Single Link Failures</u>	
<u>Links Failed</u>	<u>Throughput (KBPS/node)</u>
(1, 3)	10.8
(1, 7)	11.4
(1, 5)	10.6
(2, 3)	11.4
(2, 22)	11.5
(2, 4)	8.3
(4, 23)	9.3
(4, 9)	7.4
(4, 18)	8.4
(5, 16)	10.8
(5, 7)	9.8
(5, 19)	7.3
(6, 19)	7.8
(6, 8)	10.2
(6, 10)	6.8
(7, 23)	9.7
(8, 9)	6.5
(8, 14)	6.7
(10, 13)	7.7
(11, 17)	8.0
(11, 15)	6.8
(12, 13)	9.6
(12, 17)	9.7
(14, 21)	8.9
(15, 20)	7.4
(15, 21)	9.3
(16, 22)	11.0
(18, 20)	8.2

TWO LINK FAILURES

<u>Links Failed</u>	<u>Throughput (KBPS/node)</u>
(1, 3), (1, 5)	10.57
(1, 3), (1, 2)	10.84
(1, 3), (2, 3)	0.00
(1, 3), (2, 22)	11.00
(1, 3), (2, 4)	8.34
(1, 3), (4, 23)	9.27
(1, 3), (4, 9)	7.39
(1, 3), (4, 18)	8.40
(1, 3), (5, 16)	10.35
(1, 3), (5, 7)	9.53
(1, 3), (5, 19)	7.26
(1, 3), (6, 19)	7.81
(1, 3), (6, 8)	10.15
(1, 3), (6, 10)	6.76
(1, 3), (7, 23)	9.44
(1, 3), (8, 9)	8.49
(1, 3), (8, 14)	8.65
(1, 3), (10, 13)	7.68
(1, 3), (11, 17)	8.00
(1, 3), (11, 15)	6.76
(1, 3), (12, 13)	9.59
(1, 3), (12, 17)	9.73
(1, 3), (14, 21)	8.89
(1, 3), (15, 20)	7.30
(1, 3), (15, 21)	9.27
(1, 3), (16, 22)	10.41
(1, 3), (18, 20)	8.17
(1, 2), (1, 5)	10.57
(1, 2), (2, 3)	11.20
(1, 2), (2, 4)	8.34
(1, 2), (2, 22)	11.10
(1, 2), (4, 23)	9.27
(1, 2), (4, 9)	7.58
(1, 2), (4, 18)	8.40
(1, 2), (5, 16)	10.57
(1, 2), (5, 7)	9.81
(1, 2), (5, 19)	7.26
(1, 2), (6, 19)	7.81
(1, 2), (6, 8)	10.13
(1, 2), (6, 10)	6.76
(1, 2), (7, 23)	9.73
(1, 2), (8, 9)	8.49
(1, 2), (8, 14)	8.65
(1, 2), (10, 13)	7.68
(1, 2), (11, 17)	8.00
(1, 2), (11, 15)	6.76
(1, 2), (12, 13)	9.59

<u>Links Failed</u>	<u>Throughput (KBPS/node)</u>
(1, 2), (12, 17)	9.73
(1, 2), (14, 21)	8.89
(1, 2), (15, 20)	7.78
(1, 2), (15, 21)	9.27
(1, 2), (16, 22)	10.69
(1, 2), (18, 20)	8.59
(1, 5), (2, 3)	10.57
(1, 5), (2, 22)	9.73
(1, 5), (2, 4)	6.76
(1, 5), (4, 23)	9.16
(1, 5), (4, 9)	8.22
(1, 5), (4, 18)	8.02
(1, 5), (5, 19)	7.26
(1, 5), (5, 16)	6.76
(1, 5), (5, 7)	8.94
(1, 5), (6, 19)	7.81
(1, 5), (6, 8)	10.08
(1, 5), (6, 10)	6.76
(1, 5), (7, 23)	9.19
(1, 5), (8, 9)	8.48
(1, 5), (8, 14)	8.66
(1, 5), (10, 13)	7.68
(1, 5), (11, 17)	7.68
(1, 5), (11, 15)	6.76
(1, 5), (12, 13)	9.52
(1, 5), (12, 17)	9.01
(1, 5), (14, 21)	8.89
(1, 5), (15, 20)	6.49
(1, 5), (15, 21)	9.29
(1, 5), (16, 22)	8.00
(1, 5), (18, 20)	7.16
(2, 3), (2, 4)	8.34
(2, 3), (2, 22)	10.39
(2, 3), (4, 23)	8.46
(2, 3), (4, 9)	7.21
(2, 3), (4, 18)	8.39
(2, 3), (5, 16)	10.18
(2, 3), (5, 7)	9.77
(2, 3), (5, 19)	7.26
(2, 3), (6, 19)	7.81
(2, 3), (6, 8)	10.17
(2, 3), (6, 10)	6.76
(2, 3), (7, 23)	9.73
(2, 3), (8, 9)	8.27
(2, 3), (8, 14)	8.65
(2, 3), (10, 13)	7.68
(2, 3), (11, 17)	8.00

<u>Links Failed</u>	<u>Throughput (KBPS/node)</u>	<u>Links Failed</u>	<u>Throughput (KBPS/node)</u>
(2, 3), (11, 15)	6.76	(2, 4), (12, 17)	8.46
(2, 3), (12, 13)	9.59	(2, 4), (14, 21)	8.85
(2, 3), (12, 17)	9.73	(2, 4), (15, 20)	7.58
(2, 3), (14, 21)	8.88	(2, 4), (15, 21)	9.14
(2, 3), (15, 20)	8.28	(2, 4), (16, 22)	6.00
(2, 3), (15, 21)	9.28	(2, 4), (18, 20)	7.58
(2, 3), (16, 22)	10.94	(4, 23), (4, 18)	8.39
(2, 3), (18, 20)	8.59	(4, 23), (4, 9)	6.63
(2, 22), (2, 4)	8.34	(4, 23), (5, 16)	9.24
(2, 22), (4, 23)	8.59	(4, 23), (5, 7)	0.00
(2, 22), (4, 9)	7.03	(4, 23), (5, 19)	5.07
(2, 22), (4, 18)	8.39	(4, 23), (6, 19)	4.63
(2, 22), (5, 16)	0.00	(4, 23), (6, 8)	7.61
(2, 22), (5, 7)	9.83	(4, 23), (6, 10)	6.49
(2, 22), (5, 19)	7.26	(4, 23), (7, 23)	0.00
(2, 22), (6, 19)	7.81	(4, 23), (8, 9)	7.58
(2, 22), (6, 8)	9.81	(4, 23), (8, 14)	8.64
(2, 22), (6, 10)	6.76	(4, 23), (10, 13)	7.68
(2, 22), (7, 23)	9.73	(4, 23), (11, 17)	8.00
(2, 22), (8, 9)	8.06	(4, 23), (11, 15)	6.76
(2, 22), (8, 14)	8.64	(4, 23), (12, 13)	8.85
(2, 22), (10, 13)	7.68	(4, 23), (12, 17)	8.98
(2, 22), (11, 17)	8.00	(4, 23), (14, 21)	8.87
(2, 22), (11, 15)	6.76	(4, 23), (15, 20)	8.85
(2, 22), (12, 13)	9.59	(4, 23), (15, 21)	9.16
(2, 22), (12, 17)	9.73	(4, 23), (16, 22)	9.27
(2, 22), (14, 21)	8.87	(4, 23), (18, 20)	8.59
(2, 22), (15, 20)	7.89	(4, 9), (4, 18)	4.49
(2, 22), (15, 21)	9.29	(4, 9), (5, 16)	7.58
(2, 22), (16, 22)	0.00	(4, 9), (5, 7)	8.11
(2, 22), (18, 20)	8.59	(4, 9), (5, 19)	4.42
(2, 4), (4, 18)	7.58	(4, 9), (6, 19)	4.42
(2, 4), (4, 23)	4.63	(4, 9), (6, 8)	7.24
(2, 4), (4, 9)	6.49	(4, 9), (6, 10)	6.76
(2, 4), (5, 16)	6.76	(4, 9), (7, 23)	7.30
(2, 4), (5, 7)	5.21	(4, 9), (8, 9)	0.00
(2, 4), (5, 19)	4.86	(4, 9), (8, 14)	7.84
(2, 4), (6, 19)	4.63	(4, 9), (10, 13)	7.68
(2, 4), (6, 8)	6.80	(4, 9), (11, 17)	6.63
(2, 4), (6, 10)	6.49	(4, 9), (11, 15)	5.96
(2, 4), (7, 23)	4.86	(4, 9), (12, 13)	8.59
(2, 4), (8, 9)	7.41	(4, 9), (12, 17)	7.48
(2, 4), (8, 14)	8.62	(4, 9), (14, 21)	7.84
(2, 4), (10, 13)	7.68	(4, 9), (15, 20)	4.42
(2, 4), (11, 17)	8.00	(4, 9), (15, 21)	6.50
(2, 4), (11, 15)	6.76	(4, 9), (16, 22)	7.30
(2, 4), (12, 13)	8.50	(4, 9), (18, 20)	4.42

<u>Links Failed</u>	<u>Throughput (KBPS/node)</u>	<u>Links Failed</u>	<u>Throughput (KBPS/node)</u>
(4,18), (5,16)	7.50	(5, 7), (12,13)	8.98
(4,18), (5, 7)	8.40	(5, 7), (12,17)	8.85
(4,18), (5,19)	4.49	(5, 7), (14,21)	8.89
(4,18), (6,19)	4.42	(5, 7), (15,20)	6.87
(4,18), (6, 8)	8.06	(5, 7), (15,21)	8.97
(4,18), (6,10)	4.49	(5, 7), (16,22)	9.50
(4,18), (7,23)	8.40	(5, 7), (18,20)	7.58
(4,18), (8, 9)	4.42	(5,19), (6,19)	0.00
(4,18), (8,14)	4.49	(5,19), (6, 8)	5.21
(4,18), (10,13)	4.63	(5,19), (6,10)	6.49
(4,18), (11,17)	5.96	(5,19), (7,23)	5.43
(4,18), (11,15)	6.49	(5,19), (8, 9)	4.42
(4,18), (12,13)	4.86	(5,19), (8,14)	6.71
(4,18), (12,17)	5.43	(5,19), (10,13)	7.39
(4,18), (14,21)	4.63	(5,19), (11,17)	5.72
(4,18), (15,20)	0.00	(5,19), (11,15)	5.21
(4,18), (15,21)	4.86	(5,19), (12,13)	7.50
(4,18), (16,22)	8.40	(5,19), (12,17)	6.49
(4,18), (18,20)	0.00	(5,19), (14,21)	6.95
(5,16), (5,19)	7.26	(5,19), (15,20)	4.42
(5,16), (5, 7)	9.35	(5,19), (15,21)	6.70
(5,16), (6,19)	7.81	(5,19), (16,22)	7.26
(5,16), (6, 8)	10.13	(5,19), (18,20)	4.42
(5,16), (6,10)	6.76	(6,19), (6,10)	6.49
(5,16), (7,23)	9.25	(6,19), (6, 8)	5.72
(5,16), (8, 9)	8.49	(6,19), (7,23)	5.07
(5,16), (8,14)	8.66	(6,19), (8, 9)	4.42
(5,16), (10,13)	7.68	(6,19), (8,14)	7.06
(5,16), (11,17)	8.00	(6,19), (10,13)	7.48
(5,16), (11,15)	6.76	(6,19), (11,17)	6.49
(5,16), (12,13)	9.59	(6,19), (11,15)	5.72
(5,16), (12,17)	9.73	(6,19), (12,13)	7.40
(5,16), (14,21)	8.89	(6,19), (12,17)	7.48
(5,16), (15,20)	6.21	(6,19), (14,21)	6.99
(5,16), (15,21)	9.31	(6,19), (15,20)	4.42
(5,16), (16,22)	0.00	(6,19), (15,21)	7.26
(5,16), (18,20)	6.78	(6,19), (16,22)	7.81
(5, 7), (5,19)	5.96	(6,19), (18,20)	4.42
(5, 7), (6,19)	5.43	(6, 8), (6,10)	6.49
(5, 7), (6, 8)	8.11	(6, 8), (7,23)	7.79
(5, 7), (6,10)	6.49	(6, 8), (8, 9)	7.93
(5, 7), (7,23)	0.00	(6, 8), (8,14)	8.00
(5, 7), (8, 9)	8.27	(6, 8), (10,13)	7.68
(5, 7), (8,14)	8.66	(6, 8), (11,17)	5.96
(5, 7), (10,13)	7.68	(6, 8), (11,15)	5.21
(5, 7), (11,17)	8.00	(6, 8), (12,13)	7.68
(5, 7), (11,15)	6.76	(6, 8), (12,17)	6.76

<u>Links Failed</u>	<u>Throughput (KBPS/node)</u>	<u>Links Failed</u>	<u>Throughput (KBPS/node)</u>
(6, 8), (14, 21)	8.06	(8, 14), (15, 21)	0.00
(6, 8), (15, 20)	7.70	(8, 14), (16, 22)	8.65
(6, 8), (15, 21)	8.00	(8, 14), (18, 20)	4.63
(6, 8), (16, 22)	10.17	(10, 13), (11, 17)	0.00
(6, 8), (18, 20)	7.82	(10, 13), (11, 15)	0.00
(6, 10), (7, 23)	6.49	(10, 13), (12, 13)	0.00
(6, 10), (8, 9)	6.49	(10, 13), (12, 17)	0.00
(6, 10), (8, 14)	4.49	(10, 13), (14, 21)	4.86
(6, 10), (10, 13)	0.00	(10, 13), (15, 20)	5.21
(6, 10), (11, 17)	0.00	(10, 13), (15, 21)	5.43
(6, 10), (11, 15)	0.00	(10, 13), (16, 22)	7.68
(6, 10), (12, 13)	0.00	(10, 13), (18, 20)	4.86
(6, 10), (12, 17)	0.00	(11, 17), (11, 15)	0.00
(6, 10), (14, 21)	4.63	(11, 17), (12, 13)	0.00
(6, 10), (15, 20)	4.86	(11, 17), (12, 17)	0.00
(6, 10), (15, 21)	4.86	(11, 17), (14, 21)	6.76
(6, 10), (16, 22)	6.76	(11, 17), (15, 20)	7.68
(6, 10), (18, 20)	4.63	(11, 17), (15, 21)	7.68
(7, 23), (8, 9)	8.26	(11, 17), (16, 22)	8.00
(7, 23), (8, 14)	8.65	(11, 17), (18, 20)	6.76
(7, 23), (10, 13)	7.68	(11, 15), (12, 13)	0.00
(7, 23), (11, 17)	8.00	(11, 15), (12, 17)	0.00
(7, 23), (11, 15)	6.76	(11, 15), (14, 21)	6.76
(7, 23), (12, 13)	8.85	(11, 15), (15, 20)	6.76
(7, 23), (12, 17)	8.85	(11, 15), (15, 21)	6.76
(7, 23), (14, 21)	8.88	(11, 15), (16, 22)	6.76
(7, 23), (15, 20)	7.78	(11, 15), (18, 20)	6.76
(7, 23), (15, 21)	9.29	(12, 13), (12, 17)	0.00
(7, 23), (16, 22)	9.42	(12, 13), (14, 21)	5.43
(7, 23), (18, 20)	8.59	(12, 13), (15, 20)	5.96
(8, 9), (8, 14)	7.83	(12, 13), (15, 21)	5.96
(8, 9), (10, 13)	7.68	(12, 13), (16, 22)	9.59
(8, 9), (11, 17)	7.41	(12, 13), (18, 20)	5.43
(8, 9), (11, 15)	6.49	(12, 17), (14, 21)	5.96
(8, 9), (12, 13)	8.39	(12, 17), (15, 20)	6.76
(8, 9), (12, 17)	8.26	(12, 17), (15, 21)	6.76
(8, 9), (14, 21)	7.79	(12, 17), (16, 22)	9.73
(8, 9), (15, 20)	4.42	(12, 17), (18, 20)	5.96
(8, 9), (15, 21)	7.34	(14, 21), (15, 20)	5.43
(8, 9), (16, 22)	8.38	(14, 21), (15, 21)	0.00
(8, 9), (18, 20)	4.42	(14, 21), (16, 22)	8.88
(8, 14), (10, 13)	4.63	(14, 21), (18, 20)	4.86
(8, 14), (11, 17)	5.96	(15, 20), (15, 21)	5.96
(8, 14), (11, 15)	6.49	(15, 20), (16, 22)	6.95
(8, 14), (12, 13)	4.86	(15, 20), (18, 20)	0.00
(8, 14), (12, 17)	5.43	(15, 21), (16, 22)	9.28
(8, 14), (14, 21)	0.00	(15, 21), (18, 20)	5.43
(8, 14), (15, 20)	4.86	(16, 22), (18, 20)	7.67

TABLE 2.2.4

EFFECTS OF NODE FAILURES ON THROUGHPUT

<u>Single Node Failures</u>	
<u>Node</u> <u>Failed</u>	<u>Throughput</u> <u>(KBPS/node)</u>
1	10.44
2	8.46
3	11.06
4	4.56
5	6.27
6	6.27
7	9.76
8	7.94
9	7.65
10	7.40
11	7.71
12	9.99
13	9.35
14	8.72
15	6.54
16	10.66
17	9.75
18	8.40
19	7.81
20	8.09
21	9.03
22	11.10
23	9.52

TWO NODE FAILURES

<u>Nodes Failed</u>	<u>Throughput (KBPS/node)</u>
1,23	9.37
1, 2	0.00
1, 3	10.60
1, 4	4.66
1, 5	6.92
1, 6	6.06
1, 7	9.42
1, 8	7.96
1, 9	8.08
1,10	7.13
1,11	7.43
1,12	9.62
1,13	8.97
1,14	8.55
1,15	6.31
1,16	8.97
1,17	8.97
1,18	8.22
1,19	7.65
1,20	7.61
1,21	8.81
1,22	10.06
2,23	4.94
2, 3	9.67
2, 4	4.66
2, 5	0.00
2, 6	4.66
2, 7	5.38
2, 8	7.40
2, 9	6.13
2,10	7.13
2,11	7.43
2,12	8.35
2,13	8.20
2,14	8.50
2,15	6.31
2,16	0.00
2,17	8.81
2,18	7.97
2,19	4.94
2,20	7.96
2,21	8.87

<u>Node Failed</u>	<u>Throughput (KBPS/node)</u>
2,22	9.67
3,23	9.84
3, 4	4.66
3, 5	6.92
3, 6	6.06
3, 7	9.70
3, 8	7.92
3, 9	7.34
3,10	7.13
3,11	7.43
3,12	9.71
3,13	8.97
3,14	8.55
3,15	6.31
3,16	10.07
3,17	9.35
3,18	8.21
3,19	7.65
3,20	8.45
3,21	8.82
3,22	10.80
4,23	4.66
4, 5	0.00
4, 6	0.00
4, 7	0.00
4, 8	0.00
4, 9	4.49
4,10	4.41
4,11	4.49
4,12	4.49
4,13	4.41
4,14	4.49
4,15	0.00
4,16	4.66
4,17	4.49
4,18	4.49
4,19	0.00
4,20	0.00
4,21	4.49
4,22	4.66
5,23	0.00
5, 6	0.00

<u>Nodes Failed</u>	<u>Throughput (KBPS/node)</u>
5, 7	6.31
5, 8	4.41
5, 9	4.41
5,10	6.31
5,11	5.38
5,12	6.06
5,13	6.31
5,14	6.31
5,15	4.41
5,16	6.92
5,17	6.06
5,18	4.49
5,19	6.31
5,20	4.41
5,21	6.06
5,22	0.00
6,23	4.66
6, 7	4.94
6, 8	4.41
6, 9	4.41
6,10	7.13
6,11	0.00
6,12	0.00
6,13	0.00
6,14	4.49
6,15	0.00
6,16	6.06
6,17	0.00
6,18	4.41
6,19	6.06
6,20	4.41
6,21	4.66
6,22	6.06
7,23	9.76
7, 8	7.08
7, 9	7.82
7,10	7.13
7,11	7.43
7,12	8.81
7,13	8.97
7,14	8.55
7,15	6.31

<u>Nodes Failed</u>	<u>Throughput (KBPS/node)</u>
7,16	9.23
7,17	8.81
7,18	8.22
7,19	5.61
7,20	8.20
7,21	8.82
7,22	9.80
8,23	7.69
8, 9	7.90
8,10	4.49
8,11	5.38
8,12	4.94
8,13	4.66
8,14	7.85
8,15	0.00
8,16	7.93
8,17	5.38
8,18	4.41
8,19	4.41
8,20	4.41
8,21	0.00
8,22	7.90
9,23	7.02
9,10	7.13
9,11	6.58
9,12	6.24
9,13	8.42
9,14	7.77
9,15	4.41
9,16	7.57
9,17	7.51
9,18	4.41
9,19	4.41
9,20	4.41
9,21	7.55
9,22	7.23
10,23	7.13
10,11	0.00
10,12	0.00
10,13	8.97
10,14	4.66
10,15	0.00

<u>Nodes Failed</u>	<u>Throughput (KBPS/node)</u>
10,16	7.13
10,17	0.00
10,18	4.66
10,19	7.13
10,20	4.94
10,21	4.94
10,22	7.13
11,23	7.43
11,12	0.00
11,13	0.00
11,14	7.13
11,15	7.43
11,16	7.43
11,17	9.35
11,18	7.13
11,19	5.99
11,20	7.43
11,21	7.43
11,22	7.43
12,23	8.81
12,13	9.72
12,14	5.61
12,15	0.00
12,16	9.71
12,17	10.42
12,18	5.61
12,19	7.34
12,20	6.31
12,21	6.31
12,22	9.71
13,23	8.81
13,14	4.94
13,15	0.00
13,16	8.97
13,17	0.00
13,18	4.94
13,19	7.23
13,20	5.61
13,21	5.61
13,22	8.97
14,23	8.54
14,15	0.00
14,16	8.55

<u>Nodes Failed</u>	<u>Throughput (KBPS/node)</u>
14,17	6.31
14,18	4.66
14,19	6.92
14,20	4.94
14,21	8.82
14,22	8.54
15,23	6.31
15,16	6.31
15,17	0.00
15,18	0.00
15,19	4.41
15,20	6.31
15,21	6.31
15,22	6.31
16,23	9.18
16,17	9.35
16,18	8.08
16,19	7.65
16,20	7.20
16,21	8.81
16,22	10.66
17,23	8.97
17,18	6.31
17,19	7.13
17,20	7.43
17,21	7.43
17,22	8.97
18,23	8.22
18,19	4.41
18,20	8.58
18,21	4.94
18,22	8.22
19,23	5.15
19,20	4.41
19,21	7.16
19,22	7.65
20,23	8.58
20,21	5.61
20,22	8.33
21,23	8.83
21,22	8.83
22,23	9.89

2.3.1 Peak Throughput

Usage of the ARPA Network will differ from node to node. Generally, one can expect two kinds of users in the net--those whose peak bandwidth requirements are not very different than their average bandwidth needs and those who occasionally require very high bandwidths in relation to their average usage. The latter case includes users employing interactive graphics.

The ARPA Network as presently configured allows a typical user to enter or receive transmissions at a peak rate of about 85 KBPS if the net is not heavily loaded. Some users may never require such capacities. The question which then arises is: can average service of about 6 KBPS per node be provided to such users at a lower cost than that presently obtainable by the ARPA Network.

Average requirements of 6 KBPS per node and 2-connectivity can be supplied by installing two 9.6 KBPS links at the node requiring only low peak throughput. Since the monthly cost for such a line, \$650 plus \$0.40/mile, is significantly lower than the \$850 plus \$5.00/mile for a 50 KBPS line, one might think that considerable savings would result for a node not requiring high peak throughput. To test this hypothesis, the following experiment was performed.

The thirty nodes presently in or under consideration for the ARPA net were considered with ten additional nodes chosen from the

largest metropolitan areas not represented by the first thirty.

A low cost 40 node network was then derived using only 50 KBPS lines. This network is shown in Figure 2.1.2(g) and the nodes used are listed in Table 2.1.1. The 40 node network had a cost of \$1,025,000 and a throughput of 6.0 KBPS/node. Then, five sets of twenty nodes each were randomly selected from among the forty nodes. Each node in each set of twenty nodes was assumed to require low peak bandwidths so that these nodes could be connected into the net by either 9.6 KBPS or 50 KBPS lines, whichever was more economical. The network structure was separately optimized for each set of twenty nodes and the cost savings achieved by allowing the 9.6 KBPS lines was calculated. Finally, all forty nodes were assumed to require only low peak throughputs and the network optimization was repeated.

The results of the experiments dramatically indicated that the 9.6 KBPS line option is not generally useful for the ARPA Network. In the vast majority of cases, even when the 9.6 KBPS lines were allowed, the NAC computer optimization programs selected 50 KBPS lines for the most economical configuration. In fact, in only three cases were 9.6 KBPS lines found useful. These cases are listed in Table 2.3.1 and illustrated in Figures 2.3.1, 2.3.2, and 2.3.3. The twenty randomly selected nodes for each optimization are listed in Table 2.3.2, and the results of the optimizations, in Table 2.3.3.

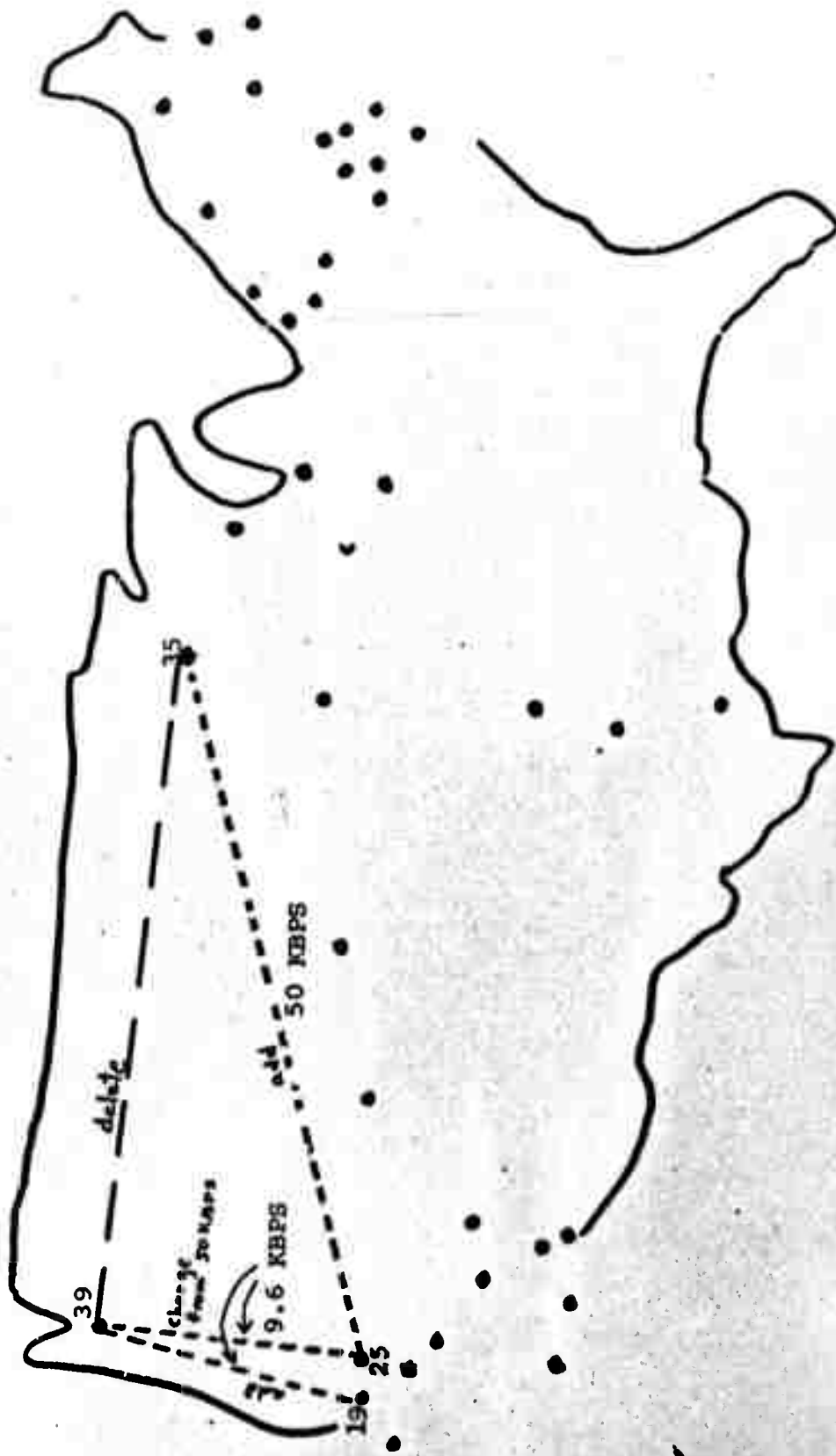


FIGURE 2.3.1

Transformation A: Save \$25,500/year



FIGURE 2.3.3

Transformation C: Save \$19,000/year

TABLE 2.3.1

<u>TRANSFORMATIONS FOR 9.6 KBS LINES</u>	
<u>Transformation</u>	<u>Applicable for 9.6 KBS at Node</u>
A	39
B	37
C	23

Throughput unaffected.


TABLE 2.3.2

TWENTY RANDOMLY SELECTED NODES FOR WHICH 9.6 KBPS LINES ARE ACCEPTABLE

<u>Group</u>	<u>Node Numbers</u>
I	1, 4, 5, 6, 9, 10, 11, 12, 15, 20, 21, 22, 23, 25, 30, 33, 34, 35, 37, 39
II.	1, 2, 3, 5, 6, 10, 11, 12, 15, 16, 18, 19, 22, 28, 31, 34, 35, 36, 37, 39
III.	1, 2, 7, 9, 11, 13, 16, 20, 21, 24, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37
IV.	1, 2, 3, 4, 5, 6, 7, 8, 11, 14, 21, 24, 25, 26, 28, 29, 33, 35, 37, 39
V.	4, 5, 7, 8, 9, 10, 11, 13, 14, 15, 17, 20, 25, 27, 28, 30, 31, 33, 35, 38
VI.	All nodes.

TABLE 2. 3. 3

NETS WITH 9.6 KBS LINES

<u>Nodes which can have 9.6 K Lines</u>	<u>Transforma- tions Used</u>	<u>Cost K\$</u>	<u>Throughput</u>
Group I	A, B, C	973.74	All throughputs for I - V, 6.0 KBS/Node 34 K Packet/Hr/Node 
Group II	A, B	992.77	
Group III	B	1,018.30	
Group IV	A, B	992.77	
Group V	None	1,025.51	
Group VI	A, B, C	973.74	
All nodes allowed (average delay 1.0 seconds)	Complete Redesign	789.72	5.72 KBS/Node 34 K Packet/Hr/Node

Note: Nets for Group I-IV all deliver average time delay no greater than 0.2 seconds.

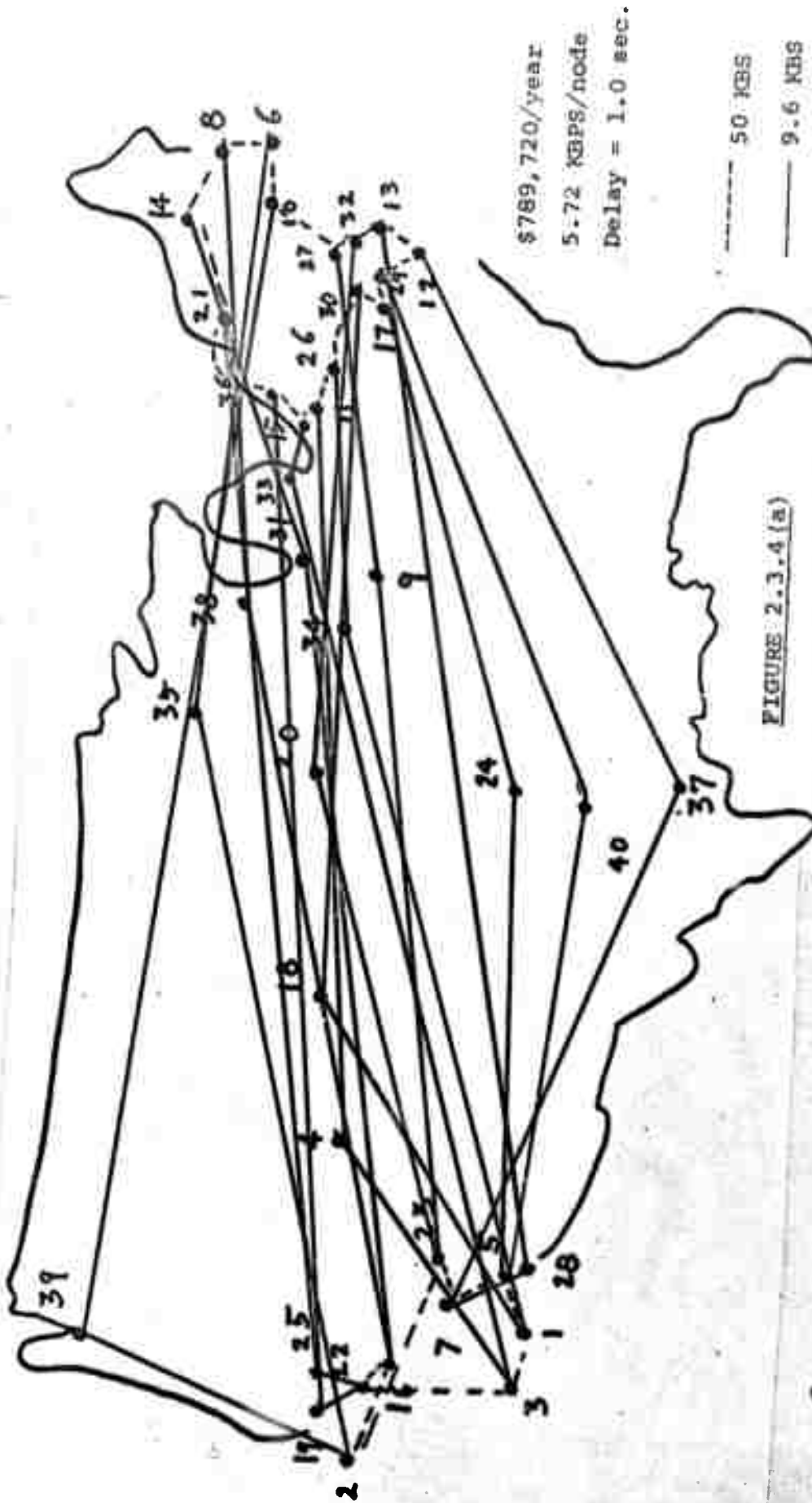


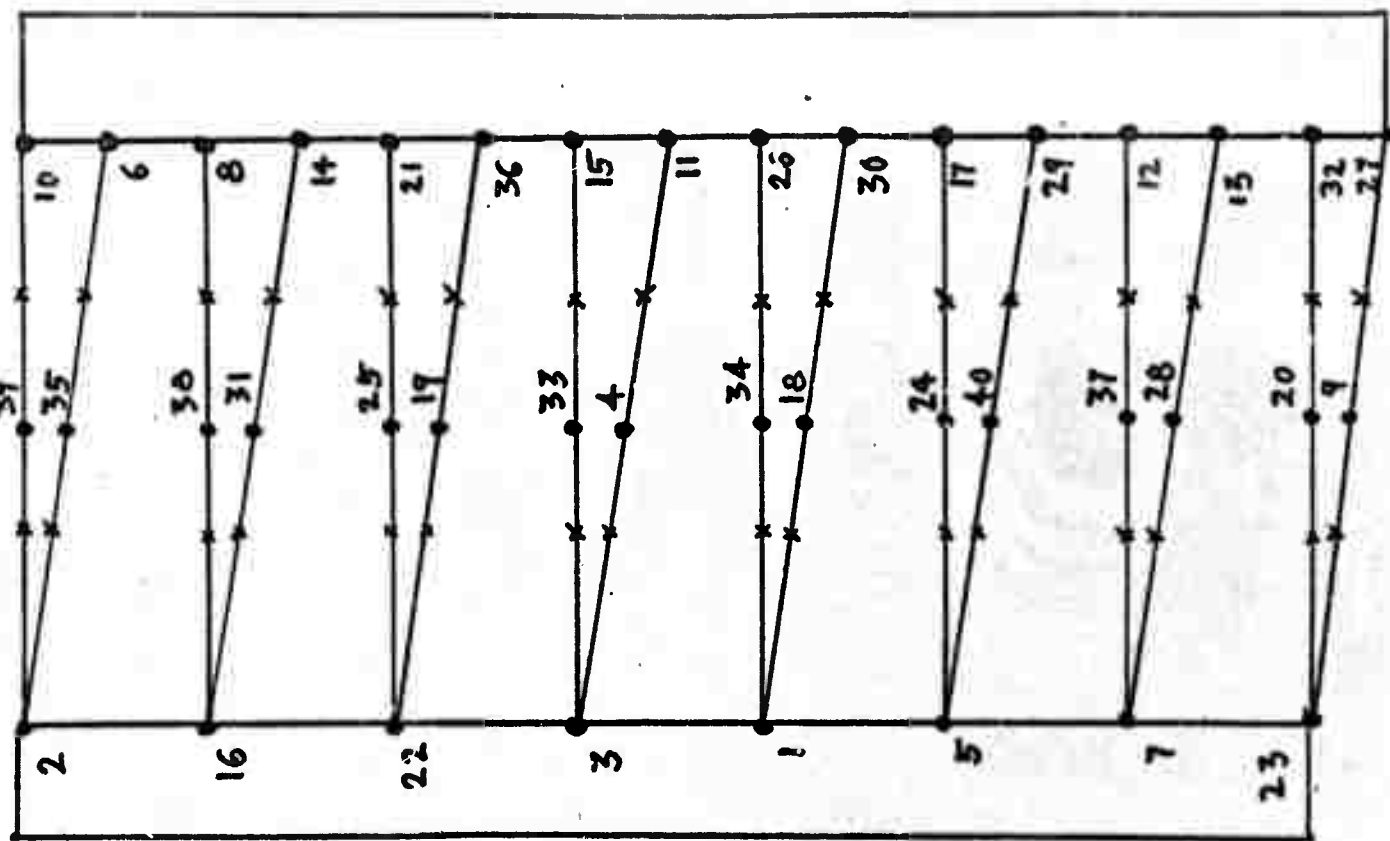
FIGURE 2.3.4 (a)

A 40-node network design providing no guaranteed high throughput capacity for any node. The net is approximately 22% less expensive than the 40-node net which guarantees high bandwidth for every node. Moreover, the net shown above has average delays of 1.0 second compared to 6.2 seconds for the 50Kbps net and it is impossible to reduce the 1.0 second delay without substantial extra cost.

\$789,720/year

5.72 KBS/Node

~34 K Packet/hr/node

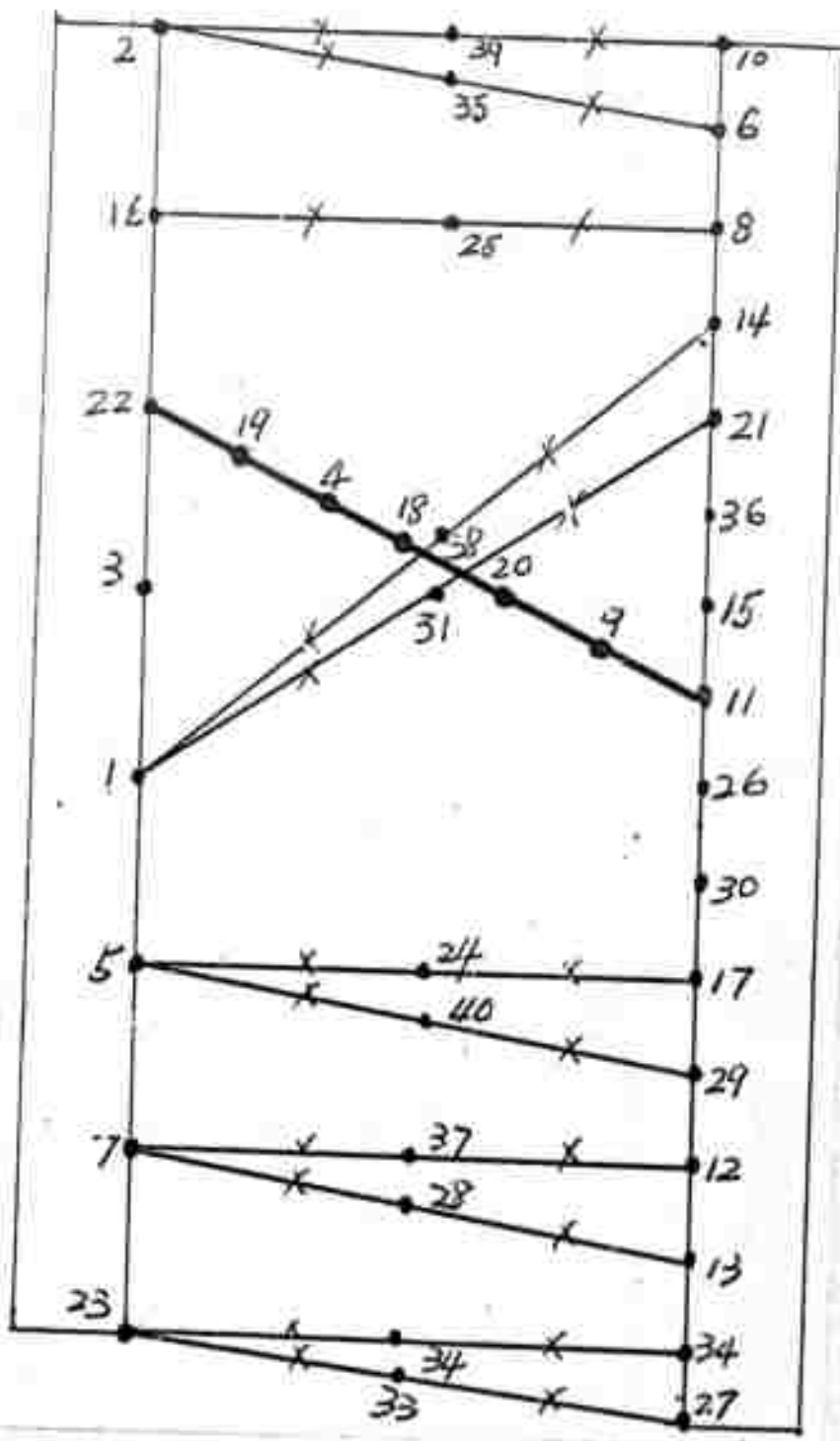


— 50 KBS

— x — 9.6 KBS

The Net in (a) redrawn.

FIGURE 2.3.4 (b)



\$854.46 K

6.0 KBS/Node

34.4 K Packets/hr/node

— 50 KBPS

— X — X — 9.6 KBPS

FIGURE 2.4.4

AN INTERESTING VARIATION

Here some nodes, judiciously selected, are connected by a 50 KBPS cross-country path yielding an average delay of .35 seconds but low peak capacities.

The yearly network line costs for the various optimizations ranged from a low of \$973,740 to \$1,025,510. The maximum savings \$51,770, which would have to be averaged over 20 nodes, represents only 10 percent of the line cost per node which itself is only approximately half the overall cost per node. Furthermore, the average savings for the entire \$1,025,000 network is less than \$25,000--a very small savings in return for a loss of high peak capacity for half the network.

The strong conclusion is that except in a few special cases (such as connecting a low requirement Seattle node), it is undesirable to use 9.6 KBPS lines in the ARPA Network.

2.3.2 Incremental Costs

The rapid growth of the ARPA Network creates the problem of equitably distributing the cost of the network over its community of users. There are two kinds of network users--ARPA contractors at universities and research centers and non-ARPA contractors who are joining the network to utilize resources such as the ILLIAC IV. The former group of users have been principally responsible for the growth and development of the network and the transition from an experimental project to a viable, economic tool broadly applicable to Defense Department communication problems. The latter user group is contributing the operating environment that will allow the network

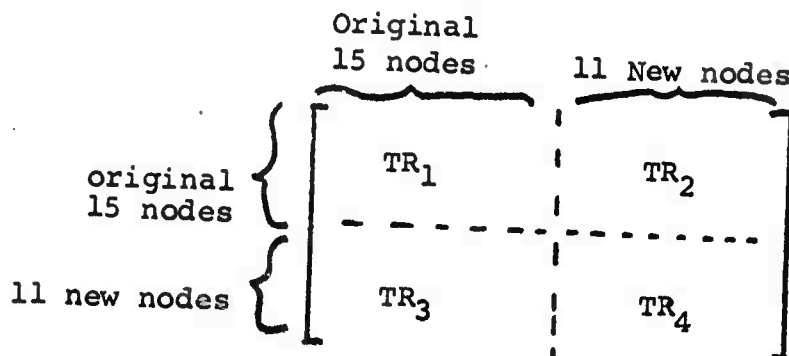
experiment to proceed from a specialized one to a general purpose one.

The problem of distributing costs between the two user groups is the subject of this section. There are several reasonable ways to attach this problem. Our approach is as follows:

1) The 15 node network shown in Figure 2.3.1 connecting the original set of ARPA contractors is first examined, and its throughput and cost determined. To evaluate network throughput, two types of analyses are used. The first assumes equal traffic between all pairs of nodes and the flow per node leading to an average time delay of 0.2 seconds is calculated. The second considers five nodes--BBN, UCLA, UCSB, SRI, and MIT--as network resources. Equal traffic from all nodes to these nodes is assumed but the return traffic from each of the five nodes to all nodes is assumed to be ten times as great as the forward traffic. The average throughput per node with this traffic pattern is then calculated at a 0.2 second time delay.

2) A 26 node network is derived by adding 11 new user nodes in the second category to the original 15 node net. The augmentation is made to minimize the incremental cost of adding the 11 nodes, without regard to throughput. The throughput of the 26-node net, which is shown in Figure 2.3.2, is calculated under a variety of

conditions. The traffic matrix $TR = [tr_{i,j}]$ where $tr_{i,j}$ is the flow from node i to node j is partitioned in the following manner.



The traffic in each of the four submatrices is adjusted independently. Using varying traffic patterns the flows in TR_1 are selected to yield a specified percentage of the full load that could be handled by the 15-node network. The maximum amount of traffic that can then be sent from the 11 new nodes is then calculated under several different assumptions about traffic patterns. In some of these calculations, NASA and NCAR are considered to be additional network resources.

3) The planned 26-node network shown in Figure 3.2.3 is analyzed in the same manner as the network discussed in 2).

The object of the three analyses is to compute the incremental cost to add the new nodes into the network, the cost required to provide service equivalent to that provided by the 15-node network to the original 15 nodes, the throughput that can be supplied to the new nodes if this should be required.

The results of the analyses are shown in Table 2.3.4

Simple conclusions that can be reached from this table are:

1) A fixed line cost of approximately \$16,500 per new node is directly attributable to the addition of the new nodes if the cost of the 15-node net is subtracted from the cost of the 26-node net.

2) Depending on the traffic pattern, the new nodes can transmit between 0 and 25 kilopackets per hour in the 26-node net if the original nodes receive throughput equal to that provided by the 15-node network. Additional throughput can only be provided to the new nodes by degrading the service to the original nodes or adding new communication links to the network. Our previous studies have shown that the incremental cost per megabit (or kilopacket) to add network capacity is about 12.5 cents if the network is ideally operated 24 hours, 7 days a week (168 hours per week). Therefore, under more typical 6 day, 12-hour service, the incremental cost per kilopacket is approximately 30 cents.

TABLE 2.3.4

All operating costs are based on a 24 hour day, 7 day week.

15 Node Net--Figure 2.3.1 Cost = \$659.0K		Approximate % of 15 Node Net Full Load	KBS/Node Between 15 Nodes	K Packets Per Hour Per Node Between 15 Nodes	KBS/Node for 11 New Nodes	K Packets Per Hr/Node For 11 New Nodes	Uniform traffic 10 units from 5 resources to all others
100	12.71	72.97	-	-	-	-	-
-	10.73	61.63	-	-	-	-	-
26 Node, Least Costly Net--Figure 2.3.2 Cost = \$759.3K		96	12.29	70.57	0	0	Uniform traffic
90	11.19	64.25	1.78	10.24			
80	10.17	58.38	3.47	19.90			
70	8.99	51.64	5.16	29.64			
50	6.57	37.73	6.57	37.73			
90	11.44	65.68	1.40	9.01			10 units from original 15 nodes to 11 new nodes other traffic=1 unit
80	10.39	59.65	2.71	15.57			
70	9.19	52.75	4.04	23.20			
45	5.68	32.62	5.68	32.62			10 units from all resources to all new other traffic=1 unit
81	10.24	58.77	0	0			10 units from 5 old resources to old nodes 10 units from 2 new resources to new nodes 1 unit of cross traffic
80	9.95	57.11	.51	2.91			
70	8.99	51.64	2.13	12.25			

Original Resources: BBN, UCLA, UCSB, SRI, MIT
New Resources: ...

TABLE 2.3.4 (Cont'd)

Approximate % of 15 Node Net Full Load	KBS/Node Between 15 Nodes	K Packets Per Hour Per Node Between 15 Nodes	KBS/Node for 11 New Nodes	K Packets Per Hr/Node For 11 new nodes
120	15.18	87.16	0	0
100	12.99	74.57	3.85	22.10
90	11.81	67.82	5.76	33.09
80	10.50	60.29	7.64	43.85
everyone equal → 74	9.44	54.21	9.14	54.21
70	9.19	52.75	9.51	54.61
Uniform traffic				
100	12.71	72.97	4.30	24.71
90	11.56	66.40	5.76	33.04
80	10.28	59.02	7.23	41.49
70	9.09	52.20	8.70	47.96
55	7.21	41.40	7.21	41.40
100	12.39	71.16	0	0
90	11.56	66.40	1.61	9.26
80	10.50	60.25	3.57	20.49
70	9.19	52.75	5.49	31.51

26 Node present network
design--Figure 2.3.3

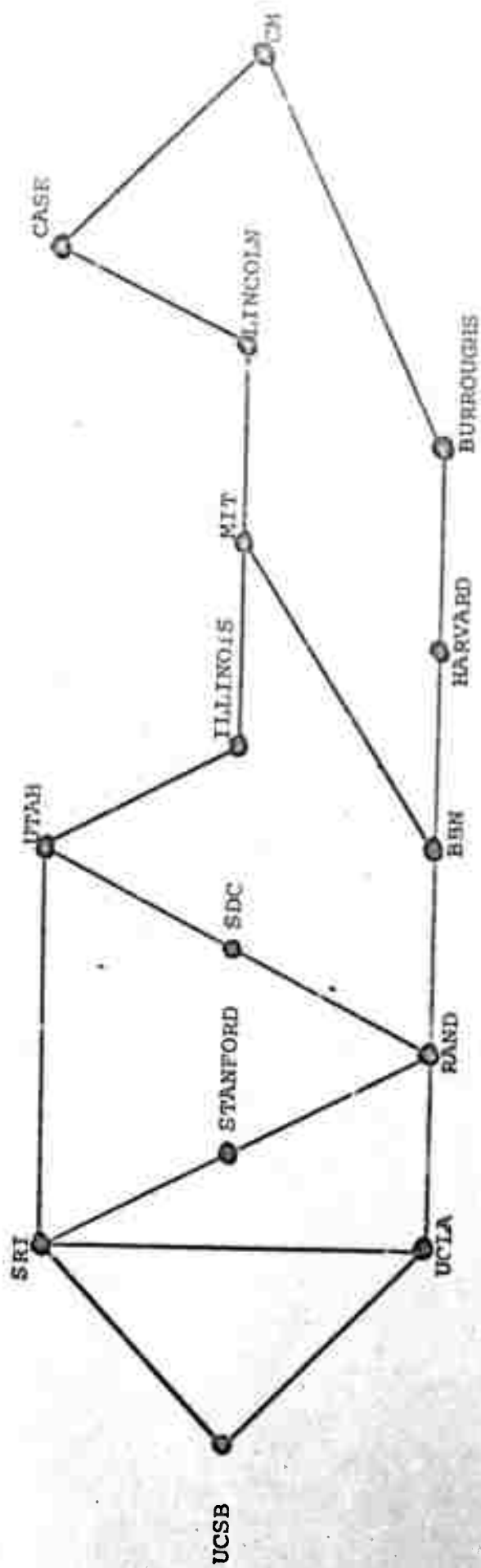


FIGURE 2.3.1

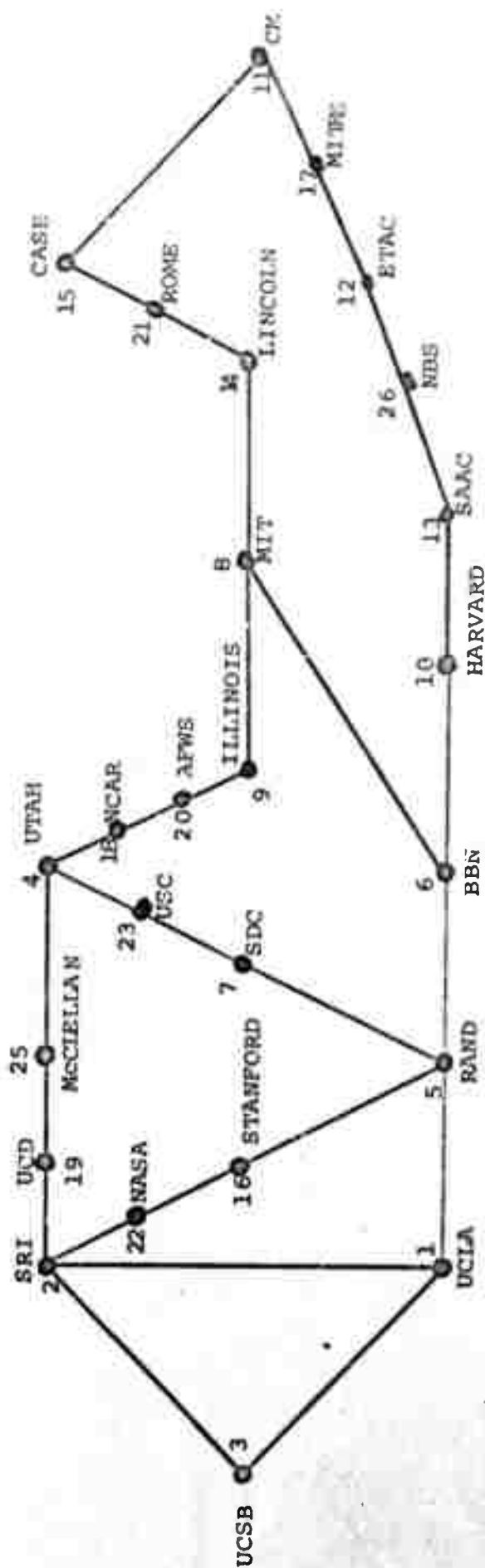


FIGURE 2.3.2

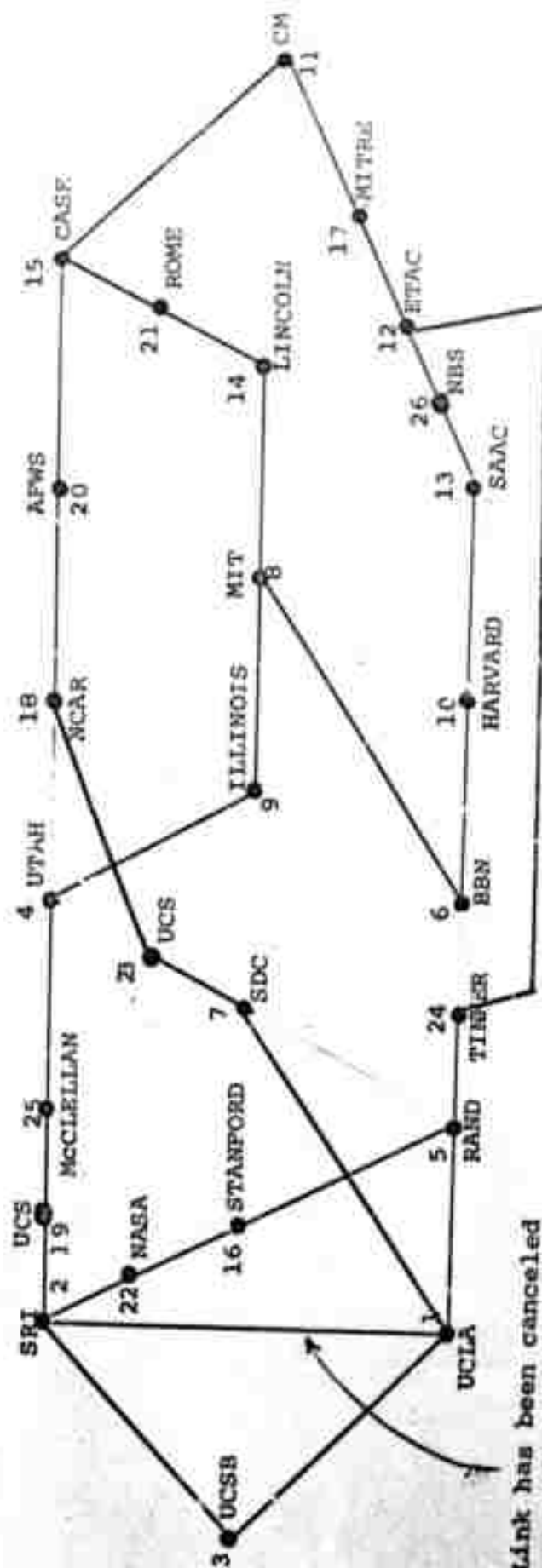


FIGURE 2.3.3

3. ROUTING STRATEGIES FOR COMPUTER NETWORK DESIGN

3.1 The Routing Problem

The routing problem in a communication network is to define a set of rules determining the path(s) over which messages should flow from one site to another. This problem is extremely complex, especially for a network of computers. A good routing procedure must be a compromise between three somewhat conflicting requirements: (1) The efficiency of any network design requires that the routing procedure make full use of available line capacities. This can be interpreted as either minimizing the average delay from message inception to arrival subject to a set of flow requirements or maximizing the throughput subject to a specified maximum delay. (2) The repeated use of the routing procedure during the design process requires it to be inexpensive to apply. Thus, it must be computationally efficient. (3) The procedure should be realistic. It should be similar to the one to be actually implemented in the final operating network and have the same general characteristics.

The objective that the delay be minimized subject to a set of flow constraints makes the routing problem a variation of a nonlinear multicommodity flow problem. This problem which was discussed in our Second Semiannual Report can be readily

formulated as a separable convex programming problem with the delay as the objective function and the conservation of flow and capacity limitations as the constraints.

The minimum delay or maximum throughput can be achieved if the routing procedure follows the solution of the programming problem. However, for networks with more than a few nodes it is not computationally efficient to solve the programming problem, and this approach is extremely expensive for repeated applications of the routing algorithm for use during the design stage.

By careful analysis of the solutions from the mathematical programming approach, it can be observed that most (over 80% in our studies) of all flow requirements are routed over paths with the minimum number of nodes. A heuristic (described in Semiannual Reports 1 and 2) deduced from this observation is to route flow over the least utilized such paths. This approach gives a result within 5%-20% of the optimum. In addition to being fast (over three orders of magnitude faster than the programming approach), it facilitates minor changes of the network structure. On the other hand, it does not take advantage of split routing, and the variations in link capacities, and leaves room for considerable improvements especially in the case when the network contains a wide distribution of different line capacities. (An apparently desirable characteristic of very large networks).

The routing procedure discussed in detail in this chapter is a heuristic one allowing split routing. Different types of flows are simultaneously routed over the paths with minimum numbers of nodes. (These paths are "shortest paths" using a simple unit metric for each line.) When no shortest path with excess capacity is available, the saturated lines are deleted from the network and flows are then routed over the shortest paths of the remaining subnetwork. The process is continued until the network is disconnected. Equally important, the algorithms used represent a major computational breakthrough for the shortest path problem. The main algorithm is based on "Floyd's algorithm" with special recognition of the fact that the node degrees in a computer network are usually low. By this technique, a message is sent down a path with fewest intermediate nodes and excess capacity, or when that path is filled, the one with next fewest intermediate nodes and excess capacity, etc. Computationwise, it is in the same order of magnitude, but faster than the first heuristic approach. Yet its results are extremely close to the optimal solution during heavy traffic. Furthermore, the routing strategy is very similar to physical routing schemes under study for the ARPA Network and it exhibits analogous behavioral properties.

3.2 A Modified Routing Algorithm for "ARPA-like" Networks

It was shown in our Second Semiannual Report that the heuristic minimum node routing strategy would yield a near optimal solution (within 5% to 20%). This heuristic algorithm described is especially efficient during the network optimization process. However, it has two drawbacks when it is used for the sole purpose of traffic routing. First, all minimum node paths between each node pair are generated while only one path is used. Computer time may be saved if only the paths to be actually used are generated. Second, the routing process terminates if any one link is saturated. Higher throughput could be obtained if alternate paths are then used.

The new routing algorithm provides the two improvements described. Based on Floyd's algorithm [3, 4], minimum node paths (hereafter called shortest paths) are generated between all node pairs. The required traffic is then routed over the unique path for each node pair. The traffic flow between each node pair and on each link is then uniformly increased or decreased until the flow is equal to the capacity for the most utilized link. The saturated link(s) is then removed from the network and the capacity on each link is replaced by its residual capacity at this point. A shortest path is again generated for each node

pair and additional traffic is routed as before. The process is repeated until the network is disconnected. Although the approach is heuristic, the result is so close to the optimal one for high traffic that no significant difference can be observed for the networks we have studied and there is no need to draw the delay curves for comparison.

Using Floyd's algorithm, the distance matrix $D = [d_{i,j}]$ and path matrix $P = [p_{i,j}]$ are defined so that $d_{i,j}$ is the distance between i and j and $p_{i,j}$ is the first intermediate node on the shortest path from i to j . Let NN be the number of nodes in the network. To generate minimum node paths, $d_{i,j}$ may be initially set equal to unity. On the other hand, it is easy to use many other path selection criteria by judiciously selecting other values of the $d_{i,j}$.

A Basic Algorithm (Floyd's Algorithm)

Step 0. Set $D = [d_{i,j}]$ and initialize the path matrix $P = [p_{i,j}]$ such that

$$d_{i,j} = \begin{cases} \text{direct distance between } i \text{ and } j \text{ if } i \text{ is} \\ \text{connected to } j. \\ \infty \text{ otherwise} \end{cases}$$

$$p_{i,j} = \begin{cases} j \text{ if } i \text{ is connected to } j \\ 0 \text{ otherwise} \end{cases}$$

Let $r = 1$.

Step 1. For $i = 1, 2, \dots, NN$ and $j = 1, 2, \dots, NN$ (If D is symmetric, the above statement is replaced by $i=1,2,\dots, NN-1$ and $j=i+1,\dots, NN$)

$$\text{let } p_{i,j} = \begin{cases} p_{i,j} & \text{if } d_{i,j} \leq d_{i,r} + d_{r,j} \\ p_{i,r} & \text{if } d_{i,j} > d_{i,r} + d_{r,j} \end{cases}$$

$$d_{i,j} = \text{Min} \{ d_{i,j}, d_{i,r} + d_{r,j} \}$$

Step 2. Stop if $r = NN$. Otherwise, let $r = r+1$ and go to Step 1.

The computation complexity of this algorithm is on the order of NN^3 , and for a long time this algorithm has been generally recognized to be the most computationally efficient general shortest path algorithm. However, its drawback is its failure to take advantage of both low degree and high degree nodes. For instance, the distance and path matrices for a tree network are easily obtained with a computation on the order of NN^2 . On the other hand, if the initial distance matrix satisfies the triangle inequality ($d_{i,k} + d_{k,j} \geq d_{i,j}$ for all i, j, k) and the network is close to a complete graph, the computation complexity is on the order of NN .

The implementation of the routing strategy given below specifically generates time savings by individually considering

high and low degree nodes. As a result, computational complexity is reduced to an order close to NN^2 for networks like the ARPA net. Detailed experience with running times is given in the next section as is a generalized algorithm for the shortest path problem.

Improved Routing Algorithm

Step 0. Initially, D and P are given as before.

Step 1. Removal of Pendant Nodes

Remove all nodes with degree of 1 and reduce the degree of their adjacent nodes. Go to Step 2 if all nodes in the network so obtained have a degree equal to 2 or greater. Stop if a node's degree is reduced to zero. (In this later case, the original network is not connected if the number of nodes removed so far is less than $NN-1$. Otherwise, it is a tree.) Repeat the process until a termination condition is met.

Step 2. Removal of Nodes with Degree Two

Let \hat{i} be any node of degree 2 and let \hat{j} and \hat{k} be its adjacent nodes. If $d_{\hat{j},\hat{k}} > d_{\hat{j},\hat{i}} + d_{\hat{i},\hat{k}}$, set $p_{\hat{j},\hat{k}} = p_{\hat{j},\hat{i}}$ and $p_{\hat{k},\hat{j}} = p_{\hat{k},\hat{i}}$. Set $d_{\hat{j},\hat{k}} = d_{\hat{k},\hat{j}} = \text{Min} \{d_{\hat{j},\hat{k}}, d_{\hat{j},\hat{i}} + d_{\hat{i},\hat{k}}\}$ and consider nodes \hat{j} and \hat{k} to be adjacent.

Step 3. Labeling Nodes with Degree Greater than Two.

3.0 Let Q be the set of nodes of degree greater than 2 and let N' be the number of such nodes.

3.1 Let i' be any element of Q

Set $A_0 = \{i'\}$ and $b_1 = i'$ and let $k = 1$ $L = 1$

3.1.1. Let $A_k = A_{k-1}$. For each j adjacent to b_k but not in A_k , let $L = L+1$, $b_L = j$, and $A_k = A_k \cup \{j\}$

3.1.2. If $A_k \neq A_{k-1}$, let $k = k+1$ and go to 3.1.1.

3.1.3. If $|A_k| \neq N'$, stop. (If $|A_k| \neq N'$, the net is not connected.) Otherwise, set $A_m = A_k$ for $m = k+1, \dots, N'$.

3.2 Define $C_i = \{h \mid h \in Q \text{ and } d_{h,i} = d_{i,h} = \infty \text{ in the initial distance matrix}\}$

For $r = 1, 2, \dots, N'$ let $r' = b_r$ and for $i \in A_k$ and $j \in C_i$ set

$$p_{j,i} = \begin{cases} p_{j,i}, & \text{if } d_{i,j} \leq d_{i,r} + d_{r',j} \\ p_{r,i} & \text{otherwise.} \end{cases}$$

$$p_{i,j} = \begin{cases} p_{i,j} & \text{if } d_{i,j} \leq d_{i,r} + d_{r',j} \\ p_{i,r'} & \text{otherwise} \end{cases}$$

and

$$d_{j,i} = d_{i,j} = \text{Min} \{d_{i,j}, d_{i,r'} + d_{r',j}\}$$

Step 4. Labeling Nodes of Degree Two

Return degree two nodes to the net in the sequence opposite to the one with which they were removed in Step 2 (i.e., last node removed is first node returned, etc.). For each node \hat{i} in this sequence with adjacent nodes \hat{j} and \hat{k} , and for any $j \in Q$, let

$$p_{j,\hat{i}} = \begin{cases} p_{j,\hat{j}} & , d_{\hat{i},\hat{j}} + d_{\hat{j},j} \leq d_{\hat{i},\hat{k}} + d_{\hat{k},j} \\ p_{j,\hat{k}} & \text{otherwise.} \end{cases}$$

$$p_{\hat{i},j} = \begin{cases} \hat{j} & \text{if } d_{\hat{i},\hat{j}} + d_{\hat{j},j} \leq d_{\hat{i},\hat{k}} + d_{\hat{k},j} \\ \hat{k} & \text{otherwise} \end{cases}$$

$$d_{j,\hat{i}} = d_{\hat{i},j} = \text{Min} \left\{ d_{\hat{i},\hat{j}} + d_{\hat{j},j} , d_{\hat{i},\hat{k}} + d_{\hat{k},j} \right\}$$

$$\text{Let } Q = Q \cup \{\hat{i}\} .$$

Step 5. Labeling Pendant Nodes

Return pendant nodes to the net in the opposite sequence to the one with which they were removed in Step 1. For each node \hat{i} in this sequence with adjacent node \hat{k} , and for any $j \in Q$, let

$$p_{i,j}^{\hat{k}} = \hat{k} \quad , \quad p_{j,\hat{i}} = p_{j,\hat{k}}$$

$$d_{j,\hat{i}} = d_{\hat{i},j} = d_{\hat{i},\hat{k}} + d_{\hat{k},j}$$

$$\text{Let } Q = U\{\hat{i}\} .$$

The above steps generate shortest paths for all node pairs. The steps separately consider degree one and two nodes. For large nets with higher average degrees, it is also possible to consider separately degree three, four, ..., nodes, with considerable computation savings. However, the average degree of a node in the ARPA net is about 2.2 and so these steps are not necessary. The next step is to determine the traffic on each link. A straightforward method is to obtain the shortest path for each node pair from the path matrix and add the flow for this node pair to each link of the path. This method would require a computation complexity of NN^3 . We therefore use the following approach which requires only NN^2 operations. In each such operation, a tree representing a combination of $NN-1$ paths is searched instead of all individual paths.

Step 6. Assign link flows

Let $BTR(a,b)$ be the flow in the link (a,b) from a to b .

For each $j = 1, 2, \dots, NN$,

6.1 Let $TR(i)$ be the required flow from node i to node j for $i = 1, 2, \dots, NN$.

6.2 Let $J = \{j_k \text{ for } k = 1, \dots, NN \text{ such that } d_{j_{k'},j} \leq d_{j_{k''},j} \text{ if } k' > k''\}$.

6.3 For $k = 1, 2, \dots, N$ and $j_k \in J$, let

$$TR(p_{j_{k'},j}) = TR(p_{j_k,j}) + TR(j_k)$$

6.4 For $i = 1, 2, \dots, NN$, set $BTR(i, p_{i,j}) = BTR(i, p_{i,j}) + TR(i)$.

Examples

The network routing procedure for store-and-forward computer nets is used in conjunction with time delay analysis models to predict levels of network traffic that can be accommodated without exceeding the appropriate delay constraints. For the ARPA net, this constraint is 0.2 seconds. The time delay analysis model is described in the Second Semiannual Report and for convenience, is summarized below.

The average time delay can be written as the sum of queueing delays over the circuits. If T_i is the mean delay time on the i -th link, then the average delay time T is

$$T = \sum_i \frac{\lambda_i}{\gamma} T_i$$

where λ_i is the total number of packets on the i -th link per unit time and γ is the total number of packets per unit time entering the network.

If C_i is the capacity of the i -th link, $1/\mu'$ the average length of a Host packet, and $1/\mu$ the average length of all packets in the system including acknowledgments, requests for next messages, headers, acknowledgments, parity checks, etc., then T can be explicitly written as

$$T = \sum_i \left\{ \frac{\lambda_i}{\gamma} \left[\frac{1}{\mu' C_i} + \frac{1}{\mu C_i} \frac{\lambda_i / \mu C_i}{1 - \lambda_i / \mu C_i} + P_i + K \right] \right\}$$

The expression

$$\frac{1}{\mu' C_i} + \frac{1}{\mu C_i} \frac{\lambda_i / \mu C_i}{1 - \lambda_i / \mu C_i} + P_i$$

represents the average time delay experienced on the i -th link by an information packet. The term

$$\frac{1}{\mu C_i} \frac{\lambda_i / \mu C_i}{1 - \lambda_i / \mu C_i}$$

is the average time an information packet spends waiting at the IMP for the i -th link to become available. Since the information packet must compete with acknowledgments and other overhead traffic, the overall message length l/μ appears in the expression. The term $l/C_i/\mu'$ is the time required to transmit an information packet of average length l/μ' . Finally, K is the nodal processing time, assumed constant and for the ARPA IMP, approximately equal to 0.3 ms; P_i is the propagation time on the i -th link (about 20 ms for a cross country link).

This formula used in conjunction with a static routing procedure to specify the link flows, is extremely useful to obtain curves of estimated time delay. For example, consider the ten node ARPA Network shown in Figure 3.2.1. Using the new routing procedure described and equal traffic requirements between all node pairs, the λ_i were found and the delay curves shown in Figure 3.2.2 were obtained. Curve A was obtained with fixed 1000 bit packets* while curve B was generated for exponentially distributed variable length packets with average size of 500 bits. In both cases, all overhead factors were ignored. Note that the delay remains small until a throughput slightly greater than 40 kilobits/second/IMP is reached. The delay then increases rapidly. Curves C and D represent the same situations

* In Case A, a generalization of the basic formula is used to allow for constant packet lengths (i.e. zero variance).

when all overhead factors are included. Notice that the throughput per IMP is reduced to 25 kilobits/second in Case C and to slightly under 20 kilobits/second in Case D.

In the same figure, we have illustrated with x's the results of a detailed simulation performed by Bolt, Beranak and Newman with a realistic routing and metering strategy. For simplicity, the simulation omitted all network overhead and assumed fixed lengths of 1000 bits for all messages. It is notable that the delay estimates from the BBN simulation (which used a dynamic routing strategy) and the computation based on the static routing strategy are in close agreement. In particular, they both accurately determined the vertical rise of the delay curve in the range just above 400 kilobits/second, the formula by predicting infinite delay and the simulation by rejecting the further input of traffic. Furthermore, the simulation did not require identical traffic from each IMP (as did the static routing procedure) and therefore the curve could actually be recomputed with a slightly skewed traffic distribution for even closer agreement.

In practice and from the analytic and simulation studies of the ARPA Network, the queueing delay is observed to remain well within the design constraint of 0.2 seconds until the

traffic within the network approaches the capacity of a cutset . The delay then increases rapidly. Assuming that alternate routing guides excess traffic along paths with excess capacity, no circuit will be operated at full capacity for very long until this point is reached and thus no sustained queuing backup should occur. Thus, as long as traffic is low enough and the routing adaptive enough to avoid the saturation of cutsets, queuing delays will not be significant.

One complete application of the new routing algorithm enables us to route along shortest paths about as much flow as the suboptimal method discussed in our Second Semiannual Report. Thus, one may expect throughputs within 5-20% of optimal and if this is acceptable (for example, during optimization) there is no need to proceed further. If, on the other hand, better results are required, saturated links may be removed one or more at a time and the complete process repeated. After each such iteration, more traffic can be sent through the network. As an example, the 23 node network shown in Figure 3.2.3 has five saturated links. Five iterations were required to obtain these flows before a cut was removed and the network was disconnected. The performance of the network after each iteration is plotted as throughput versus delay curves in Figure 3.2.4.

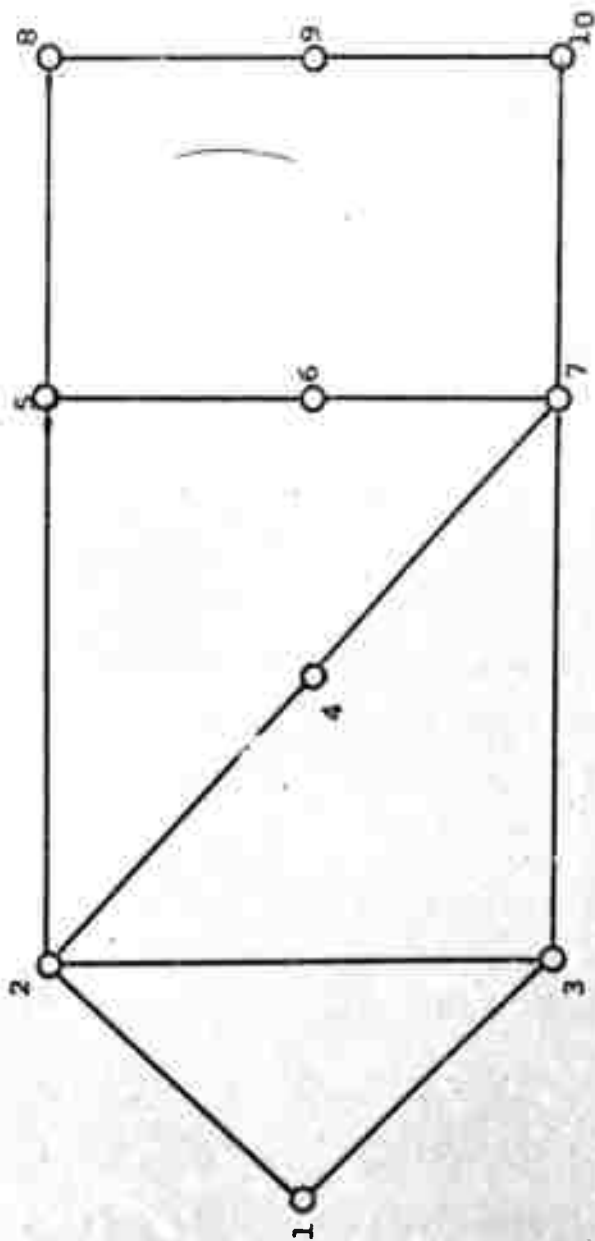


FIGURE 3.2.1

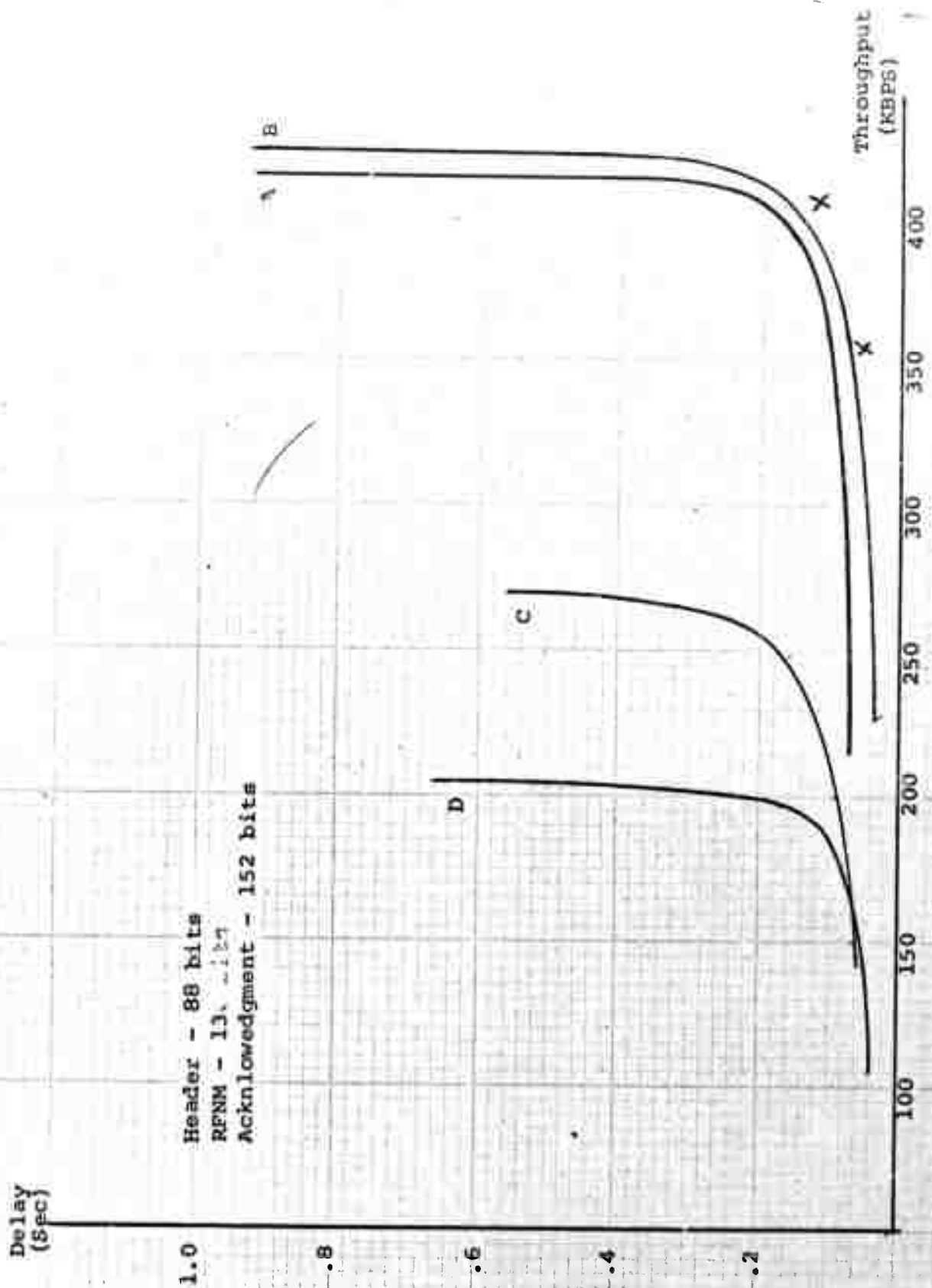


FIGURE 3.2.2

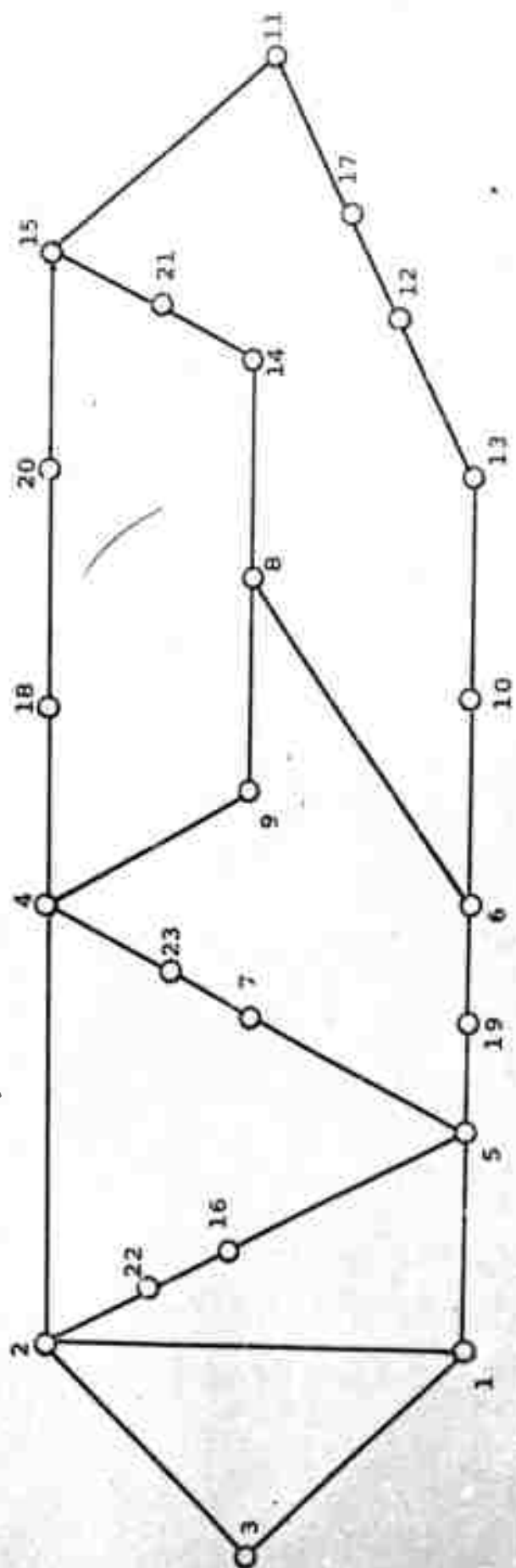


FIGURE 3.2.3

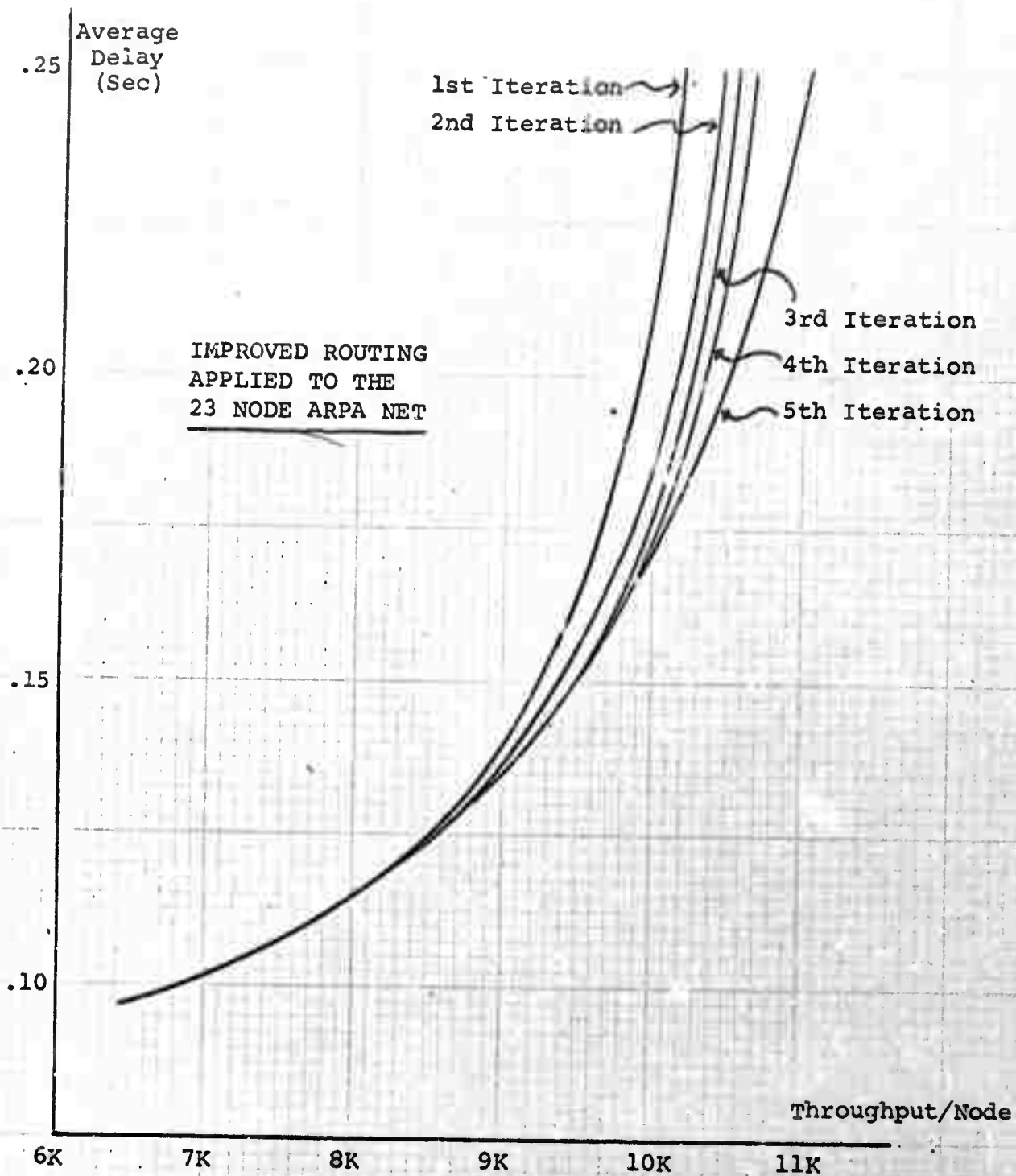


FIGURE 3.2.4

3.3 New Shortest Route Algorithms

NAC's improved shortest route algorithm consists of two parts: a modified version of Floyd's algorithm and a node truncation step. Step 3 of Section 3.2 represents a modified version of Floyd's algorithm. This version is especially efficient for, but not limited to, networks satisfying the triangle inequality. A basic step for the ordinary Floyd's algorithm is to set $d_{i,j} = \text{Min} \{ d_{i,k} + d_{k,j}, d_{i,j} \}$ for all k, i, j . This equation performs unnecessary additions and comparisons for those j 's which are directly adjacent to i , and for those j 's and i 's which do not have a path to k with a finite value. In our modification, we only choose those j 's which do not neighbor i , and thereby savings on computation can be made for highly connected networks. Furthermore, we only choose i 's and j 's which have a finite path to k ; thereby we can save computing time for sparsely connected networks.

Steps 1, 2, 4 and 5 in Section 3.2 represent the node truncation part. The purpose of the truncation is to reduce the size of the network being considered so that computation can be simplified when either the ordinary or the modified Floyd's algorithm is applied on the remaining network. As already stated, the node truncation technique is not limited to nodes of degree one and two as long as the savings on applying Floyd's algorithm warrants the extra overhead involved in the truncation. For

efficient use of the algorithms, we need to know at which level of node degrees, the node truncation approach is no longer advantageous. We will discuss two cases.

The General Case

In this case link costs (lengths) may assume any value as long as there is no circuit whose sum of costs is negative. Symmetry of the link cost matrix is not necessary.

Let NN be the number of nodes in the network and D the degree of the node to be truncated. The number of additions and comparisons required at the time the node is removed from the net is equal to $D(D-1)$. The number of additions and comparisons required at the time the node is returned to the net is $2(D-1)(NN-1)$. The total number of additions and comparisons required for the node truncation is therefore equal to $(D-1) \cdot$

$(2NN + D - 2)$. The number of additions and comparisons saved if Floyd's algorithm is then applied to the remaining net is $NN(NN-1)(NN-2) - (NN-1)(NN-2)(NN-3) = 3(NN-1)(NN-2)$.

Therefore, the net saving on the number of additions and comparisons is: $3(NN-1)(NN-2) - (D-1)(2NN - D - 2)$. This number is positive for $D < NN-1$ and zero if $D = NN-1$. Therefore, except for the limiting case of a completely connected net, the node truncation technique is always faster than Floyd's method. The algorithm

for handling the general case is given below.

(1) GENERAL NODE TRUNCATION SHORTEST PATH ALGORITHM

Step 0. Let $K = 1$.

Step 1. Removal of pendant nodes

1.1 If there are no degree one nodes, go to Step 2.

1.2 Pick any degree one node and renumber it as v_K . Let B_K be its adjacent node, and reduce the degree of B_K by one. Let $K = K+1$ and go to 1.1.

Step 2. Removal of nodes with degree greater than one.

2.1 Rename the remaining nodes as $v_{NN}, v_{NN-1}, \dots, v_K$ such that their degrees $D_{NN} \geq \dots \geq D_K$. Let $H = NN-K+1$.

2.2 Let $B_K = \{K_1, K_2, \dots, K_{q(K)}\}$ be the sets of nodes adjacent to v_K in the remaining network, where $q(K) = \min \{D_K, H-1\}$.

2.3 For $i, j = 1, 2, \dots, q(K)$, set

$$d_{K_i, K_j} = \min \{d_{K_i, K_j}, d_{K_i, v_K} + d_{v_K, K_j}\}$$

$$p_{K_i, K_j} = \begin{cases} p_{K_i, K_j} & \text{if } d_{K_i, K_j} \leq d_{K_i, v_K} + d_{v_K, K_j} \\ p_{K_i, v_K} & \text{otherwise} \end{cases}$$

$$D_{K_i} = D_{K_i} + 1 \text{ if } K_i \text{ is not directed to } K_j$$

$$D_{K_j} = D_{K_j} + 1 \text{ if } K_j \text{ is not directed to } K_i$$

$K_1, K_2, \dots, K_{q(K)}$ are considered to be adjacent to each other.

2.4 If $H = 2$, go to Step 3. Otherwise let $H = H-1$ and $K = K+1$, and go to 2.2.

Step 3. Labeling

3.1 For $j = K+1, \dots, NN$, set

$$d_{j,v_K} = \min_{K_i \in B_K} \{d_{j,K_i} + d_{K_i,v_K}\}$$

$$p_{j,v_K} = p_{j,h} \text{ where } h \text{ satisfies } d_{j,v_K} = d_{j,h} + d_{h,v_K}$$

$$d_{v_K,j} = \min_{K_i \in B_K} \{d_{v_K,K_i} + d_{K_i,j}\}$$

$$p_{v_K,j} = p_{v_K,r} \text{ where } r \text{ satisfies } d_{v_K,j} = d_{v_K,r} + d_{r,j}$$

3.2 If $K = 1$, Stop. Otherwise, let $K = K-1$ and go to 3.1

The Special Case: Symmetry and triangle inequality are satisfied.

It can be shown that regardless of the exact degree of any node the node truncation approach is always faster than the ordinary Floyd's algorithm. However, the modified version described earlier of Floyd's algorithm may be faster for nodes with large degrees. The exact values depend on the topology of the network, and there is no simple formula to determine the breakpoint. We evaluate the tradeoffs between the two approaches as follows. We calculate a node degree DT such that if $D < DT$ where D is the degree of any node during the labeling process, then it is computationally more efficient to remove the node via truncation than to consider the node during labeling by the revised Floyd's algorithm.

Assume that the modified Floyd's algorithm is to be applied on the network remaining after node k is removed. Let DC_i be the number of nodes not adjacent to node i in the starting network.

Let D be the degree of node k and let

$$TCD = \sum_{i=1}^{NN} CD_i$$

The net saving on the number of additions and comparisons is more than

$$\begin{aligned} & \frac{1}{2} [TCD(NN-2) - (TCD - 2CD_k)(NN-d)] - \frac{1}{2} (D-1)(2NN+D-2) \\ &= \frac{1}{2} [TCD + 2CD_k(NN-3)] - \frac{1}{2} (D-1)(2NN+D-2) \\ &\geq CD_k(NN-3) - \frac{1}{2} (D-1)(2NN+D-2) \\ &\geq (NN-3)(NN-D-1) - \frac{1}{2} (D-1)(2NN+D-2) \\ &= \frac{1}{2} (-D^2 - (4NN-7)D + 2(NN-1)^2) > 0 \text{ for } D < .42 NN \end{aligned}$$

Therefore, for any $D < .42 NN$ it is always better to use truncation. For the special case being considered (i.e. a symmetric distance matrix which satisfies the triangle inequality) the algorithm may be stated as given below.

(2) SPECIAL NODE TRUNCATION SHORTEST PATH ALGORITHM

Step 0. Same as in general case.

Step 1. Same as in general case.

Step 2.

2.1 Same as in general case.

2.2 If $D > 0.42 NN$, go to Step 3.

2.3 Let $B_K = K_1, K_2, \dots, K_{D_K}$ be the sets of nodes adjacent to v_K in the remaining network. Let E_K be a subset of B_K which contains nodes adjacent to v_K in the starting network.

For $i=1, 2, \dots, D_K - 1$ and $j = i+1, \dots, D_K$ set

$$d_{K_i, K_j} = d_{K_j, K_i} = \text{Min} \{ d_{K_i, K_j}, d_{K_i, v_K} + d_{v_K, K_j} \}$$

$$p_{K_i, K_j} = \begin{cases} p_{K_i, K_j} & \text{if } d_{K_i, K_j} \leq d_{K_i, v_K} + d_{v_K, K_j} \\ p_{K_i, v_K} & \text{otherwise.} \end{cases}$$

$$p_{K_j, K_i} = \begin{cases} p_{K_j, K_i} & \text{if } d_{K_i, K_j} \leq d_{K_j, v_K} + d_{v_K, K_i} \\ p_{K_j, v_K} & \text{otherwise} \end{cases}$$

$D_{K_i} = D_{K_j} = D_{K_i} + 1$ if K_i and K_j are not adjacent nodes.

K_1, K_2, \dots, K_{D_K} are considered to be adjacent to each other in the remaining network.

- 2.4 Let $H = H - 1$ and $K = K+1$. If $H = 2$, go to Step 3.
Otherwise, go to 2.2.

Step 3.

- 3.0 Let $Q = \{v_{NN}, v_{NN-1}, \dots, v_K\}$, $N' = NN-K+1$
3.1 Same as in Section 3.2.
3.2 Same as in Section 3.2.

Step 4. Labeling

- 4.0 $K = K-1$
4.1 For any $j \in \{K+1, \dots, N\} = E_K$, set

$$d_{j,v_K} = d_{v_K,j} = \min_{K_i \in B_K} \{d_{j,K_i} + d_{K_i,v_K}\}$$

$$p_{j,v_K} = p_{j,h}; \quad p_{v_K,j} = p_{v_K,h},$$

where h satisfies $d_{j,v_K} = d_{j,h} + d_{h,v_K}$

- 4.2 If $K=1$, Stop. Otherwise, let $K = K-1$ and go to Step 4.1.

Even though we have only considered the shortest path problems for all node pairs, the node truncation techniques can also be used to determine shortest paths between specified pairs of nodes.

- 2.4 Let $H = H - 1$ and $K = K+1$. If $H = 2$, go to Step 3.
Otherwise, go to 2.2.

Step 3.

- 3.0 Let $Q = \{v_{NN}, v_{NN-1}, \dots, v_K\}$, $N' = NN-K+1$
3.1 Same as in Section 3.2.
3.2 Same as in Section 3.2.

Step 4. Labeling

- 4.0 $K = K-1$
4.1 For any $j \in \{K+1, \dots, N\} = E_K$, set

$$d_{j,v_K} = d_{v_K,j} = \min_{K_i \in B_K} \{d_{j,K_i} + d_{K_i,v_K}\}$$

$$p_{j,v_K} = p_{j,h}; \quad p_{v_K,j} = p_{v_K,h},$$

where h satisfies $d_{j,v_K} = d_{j,h} + d_{h,v_K}$

- 4.2 If $K=1$, Stop. Otherwise, let $K = K-1$ and go to Step 4.1.

Even though we have only considered the shortest path problems for all node pairs, the node truncation techniques can also be used to determine shortest paths between specified pairs of nodes.

3.4 Computational Comparisons

The ordinary Floyd's algorithm, NAC's shortest route algorithm, and traffic assignment algorithm have been applied to various 10 node, 20 node, 30 node, 40 node, and 80 node networks. The computing times are recorded and are listed on Table 3.4.1. ARPA-like and ARPA networks for these experiments are shown on Figure 3.4.1, Figure 3.4.2, Figure 3.4.3 and Figure 2.1.2(g). Minimum spanning trees are based on ARPA nodes and are given on Figure 3.4.4 to Figure 3.4.7. Minimum link x -connected graphs [1] are those that are x -connected and are constructed with minimum number of links. (A graph is x -connected if it requires the removal of at least x nodes to disconnect it). Minimum link 2, 3, 4 and 5 connected 10-node graphs are shown on Figure 3.4.8. Those with more nodes are formed in a similar way. Data from Table 3.4.1 are graphed as a function of computing time versus number of nodes in Figure 3.4.9. These curves clearly indicate that computational complexity is proportional to NN^3 for ordinary Floyd's algorithm regardless of the topology and is proportional to NN^2 for NAC's algorithm when applied to telecommunication networks (i.e. tree structures, ARPA-like networks, 2-connected networks, etc.), and is proportional to NN^2 for traffic assignment. For moderate (most nodes with degrees higher than 2) and high connected networks, the node

truncation part is not used in these figures since it was not felt that programming this general algorithm would be immediately necessary for our ARPA net studies. The NAC modified version of Floyd's algorithm, even when used alone outperforms the ordinary one, especially for highly connected nets. The main advantage the modified version has over the ordinary one is its use of the triangle inequality. Otherwise, its efficiency would not be very much better than that of the original Floyd's algorithm. Second, since the NAC algorithm requires larger overhead, the larger the net, the better efficiency the NAC algorithm shows. As can be observed from Fig. 3.4.7 for some 10 node nets, NAC's algorithm may even be less efficient due to the overhead. (The lower ends of the curves are not quite proportional to NN^2 or N^3 rules because of overhead factors which are significant when the net is small.)

TABLE 3.4.1

COMPUTATION TIMES FOR SHORTEST PATH ALGORITHMS
(In Milliseconds)

Network		Label Process		Traffic Assignment
		Floyd's Algo.	NAC's Algo	
10 Nodes	ARPA Net (Fig. 3.4.1)	.010	.008	.007
	Minimum Spanning Tree (Fig. 3.4.4)	.010	.002	.008
	Minimum Link 2-connected	.010	.007	.008
	Minimum Link 3-connected	.010	.011	.008
	Minimum Link 4-connected	.009	.010	.008
	Minimum Link 5-connected	.009	.011	.007
	Minimum Link 8-connected	.010	.009	.007
20 Nodes	ARPA Net (Fig. 3.4.2)	.064	.020	.023
	Minimum Spanning Tree (Fig. 3.4.5)	.062	.008	.025
	Minimum Link 2-connected	.063	.018	.024
	Minimum Link 3-connected	.064	.052	.025
	Minimum Link 4-connected	.063	.045	.023
	Minimum Link 5-connected	.062	.056	.023
	Minimum Link 18-connected	.063	.031	.019
30 Nodes	ARPA Net (Fig. 3.4.3)	.211	.034	.055
	Minimum Spanning Tree (Fig. 3.4.6)	.209	.015	.055
	Minimum Link 2-connected	.209	.033	.057
	Minimum Link 3-connected	.211	.142	.056
	Minimum Link 4-connected	.210	.131	.054
	Minimum Link 5-connected	.212	.160	.054
	Minimum Link 28-connected	.208	.063	.040
40 Nodes	ARPA Net (Fig. 2.1.2(h))	.501	.057	.098
	Minimum Spanning Tree (Fig. 3.4.7)	.510	.028	.100
	Minimum Link 2-connected	.501	.056	.102
	Minimum Link 3-connected	.511	.298	.099
	Minimum Link 4-connected	.508	.276	.097
	Minimum Link 5-connected	.508	.334	.095
	Minimum Link 38-connected	.501	.102	.070
80 Nodes	Minimum Link 2-connected	4.075	.224	.380
	Minimum Link 4-connected	3.975	2.141	.388
	Minimum Link 10-connected	3.968	2.329	.377
	Minimum Link 40-connected	4.020	2.359	.341
	Minimum Link 78-connected	4.097	.467	.293

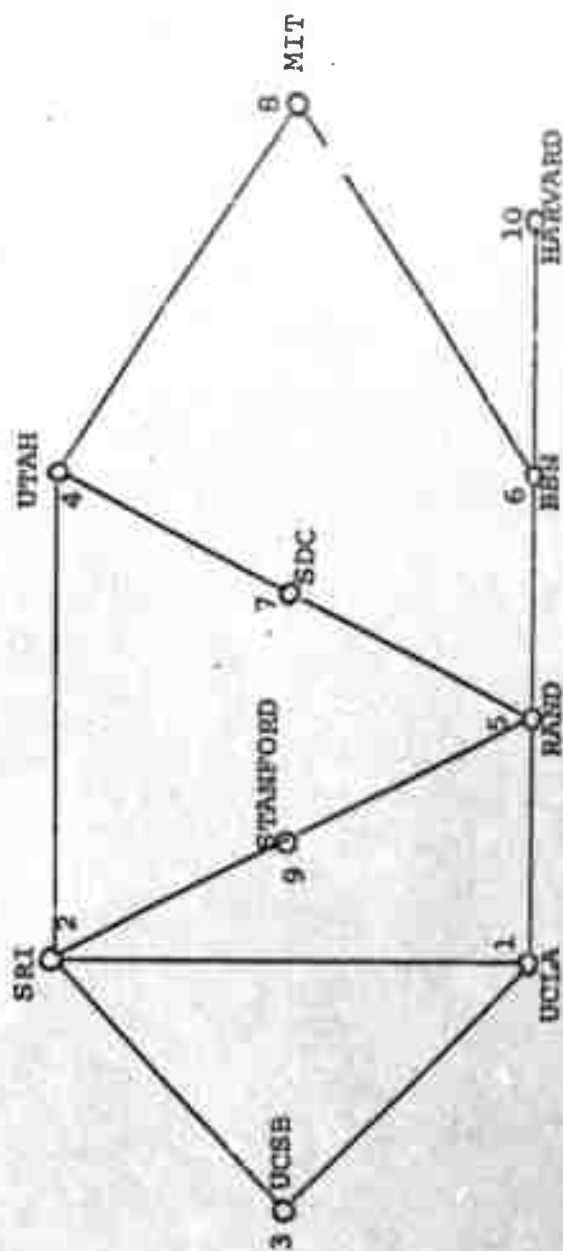


FIGURE 3.4.1

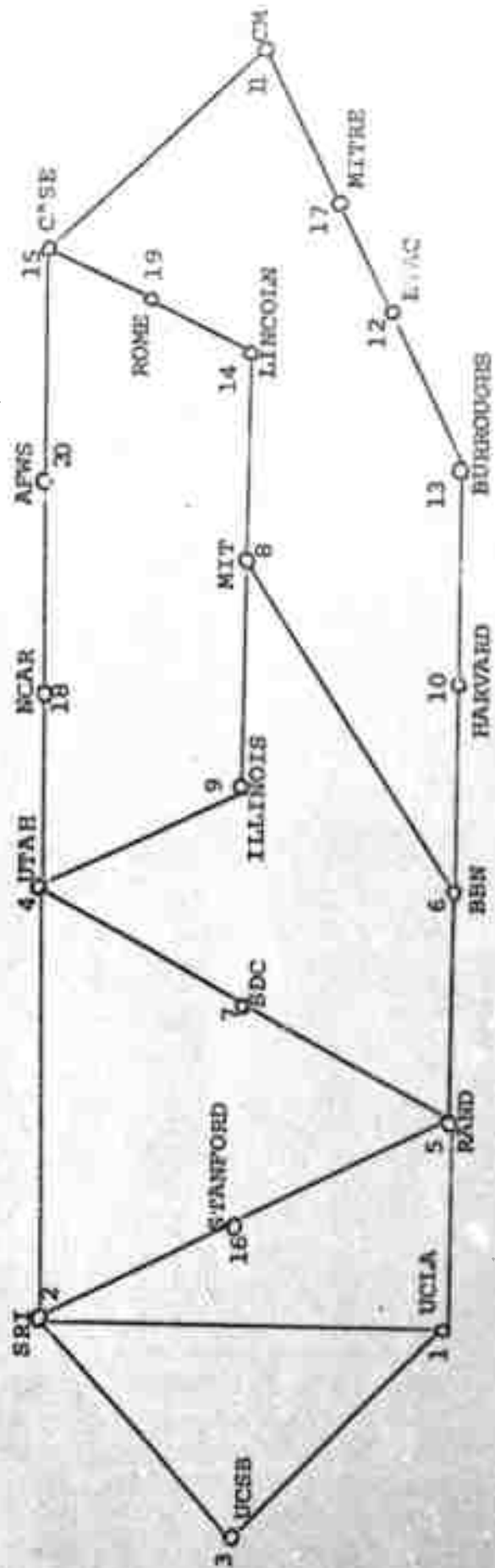


FIGURE 3.4.2

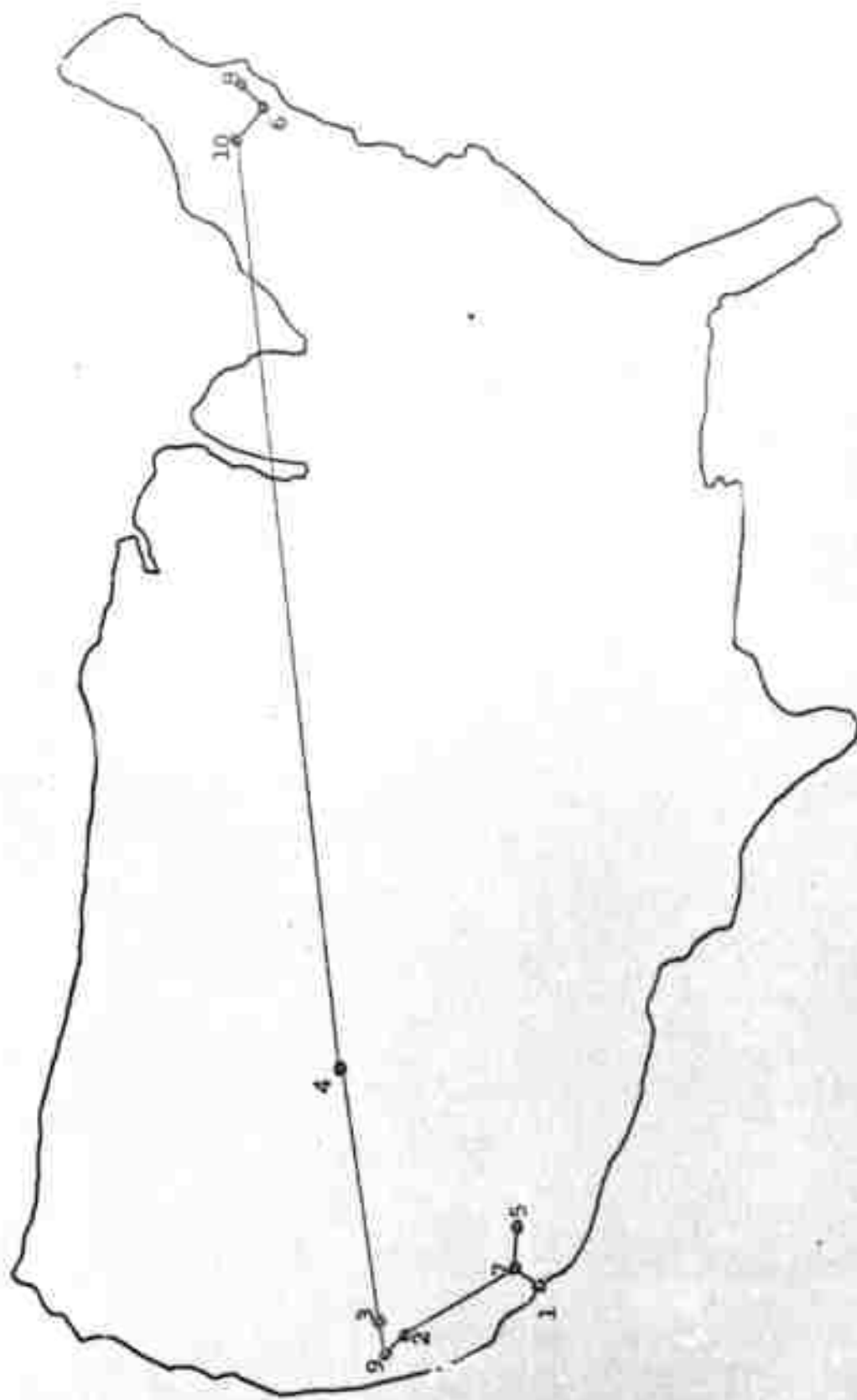


FIGURE 3.4.4

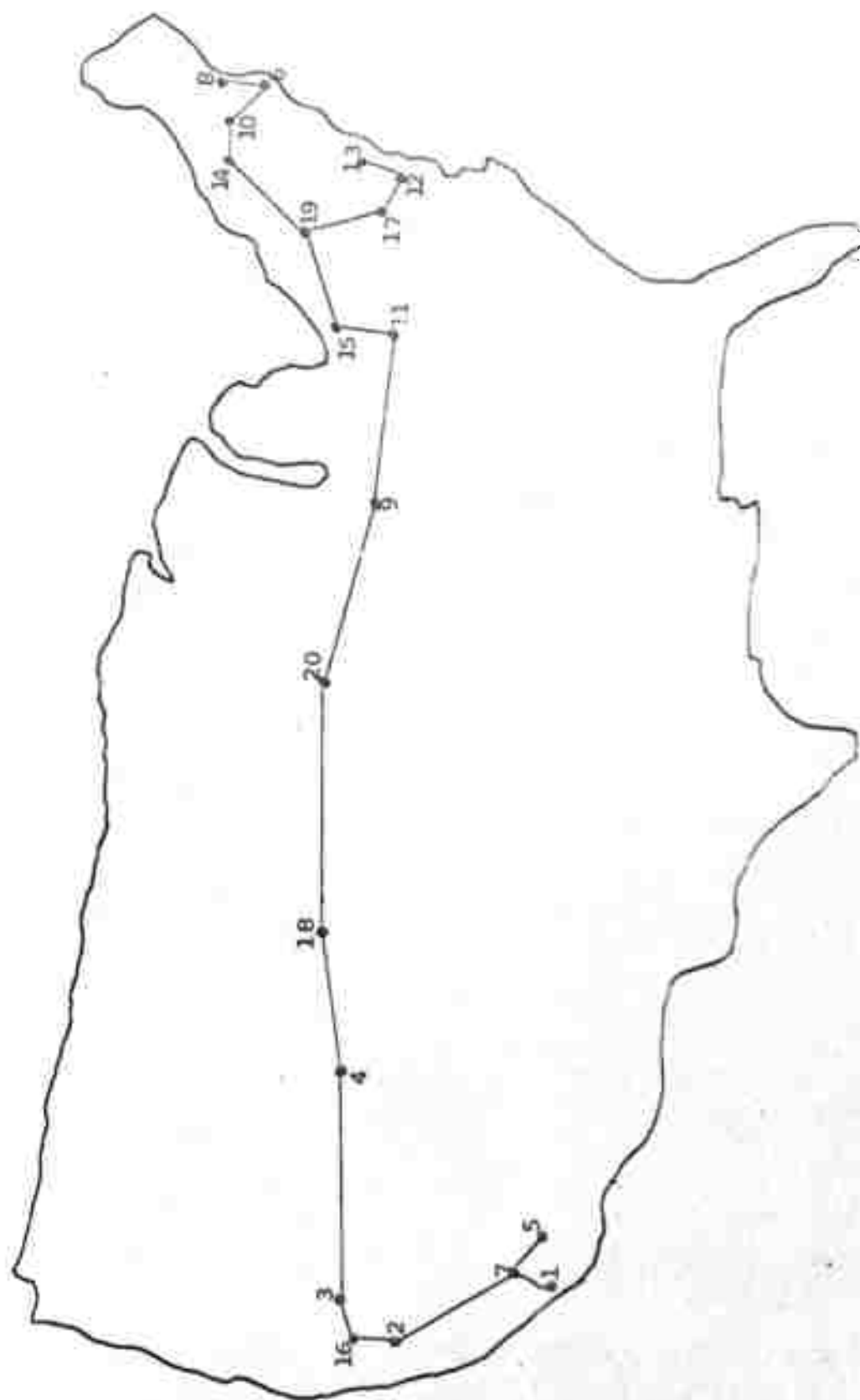


FIGURE 3.4.5

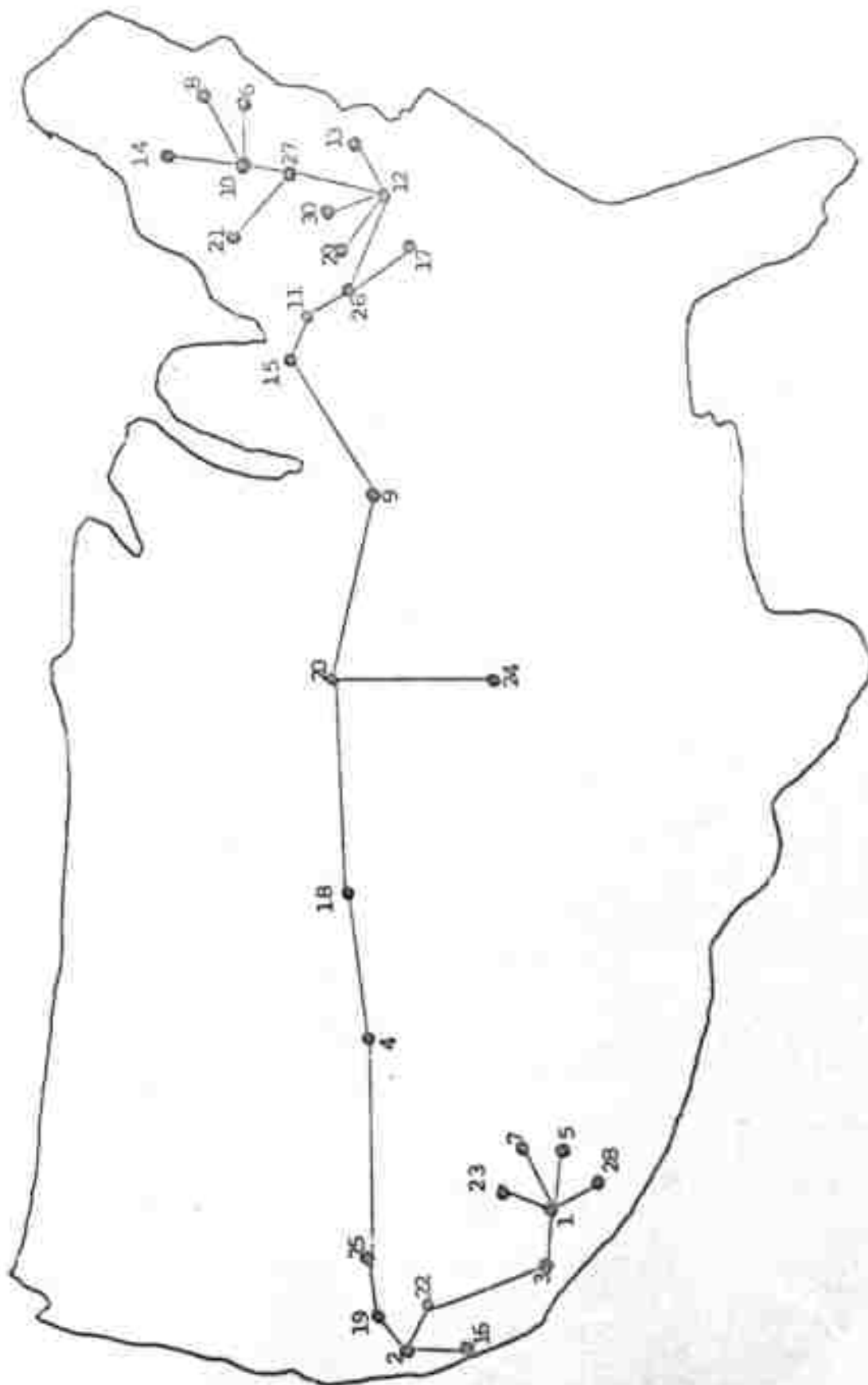


FIGURE 3.4.6

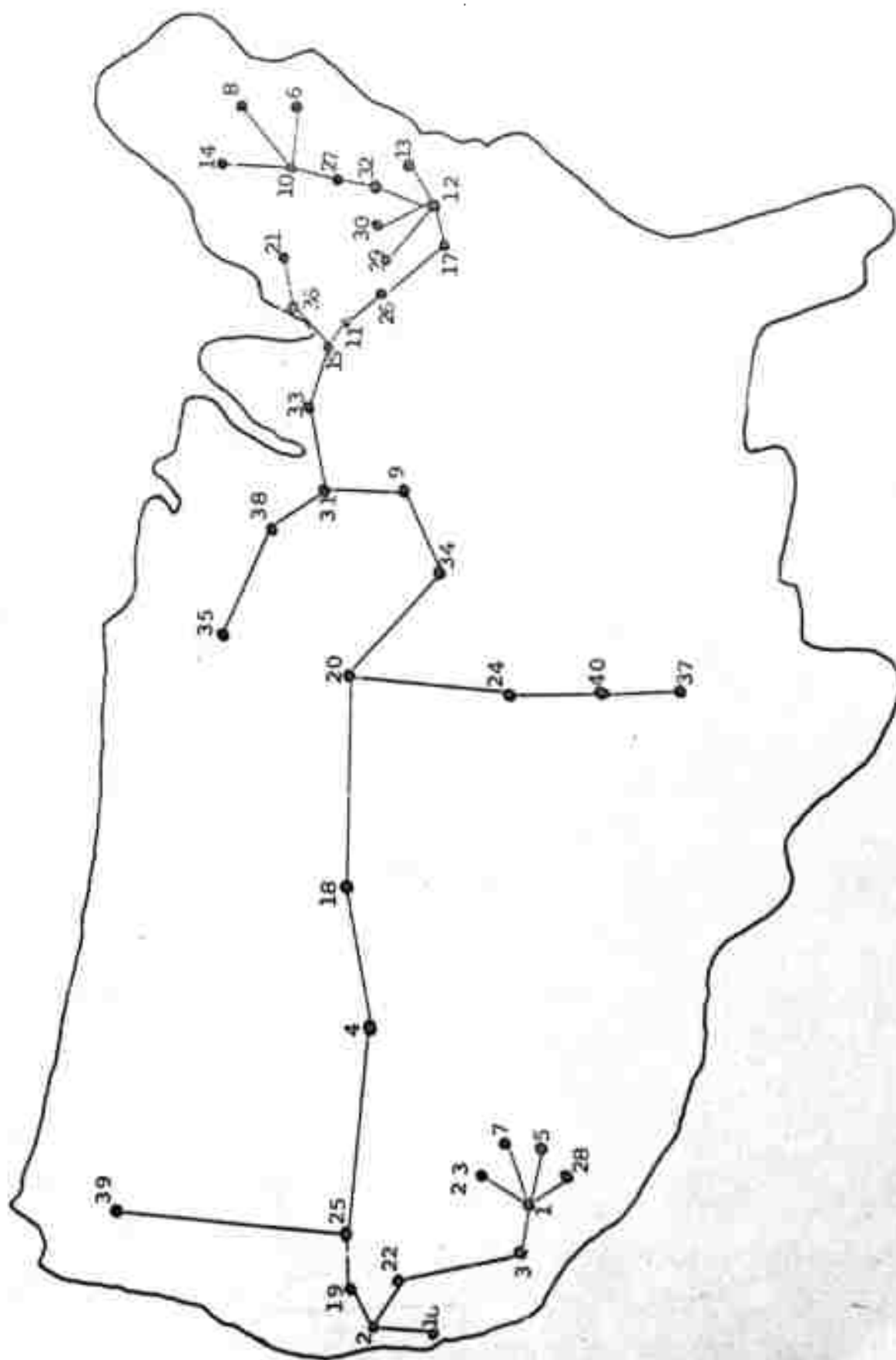
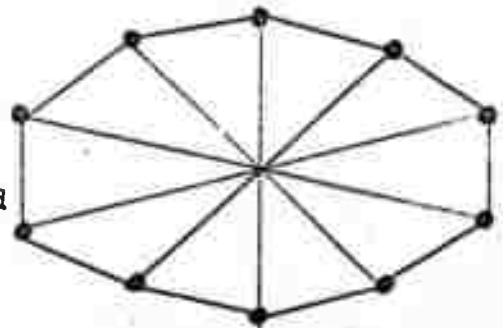


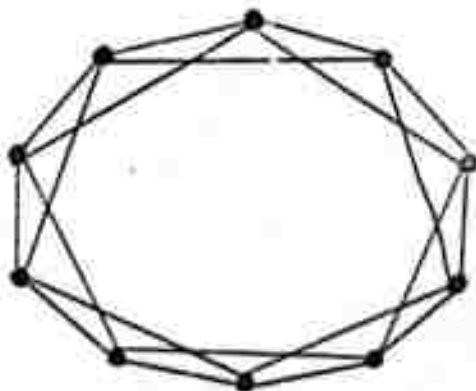
FIGURE 3.4.7



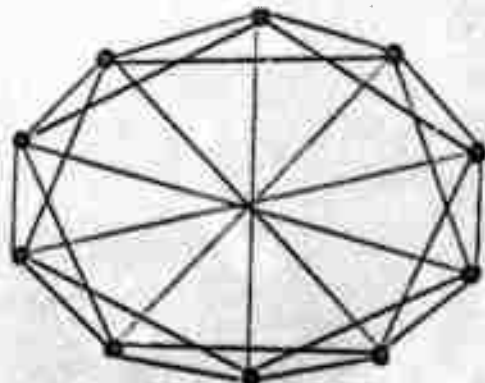
Minimum Link 2 Connected



Minimum Link 3 Connected



Minimum Link 4 Connected



Minimum Link 5 Connected

10 NODE NETS

FIGURE 3.4.8

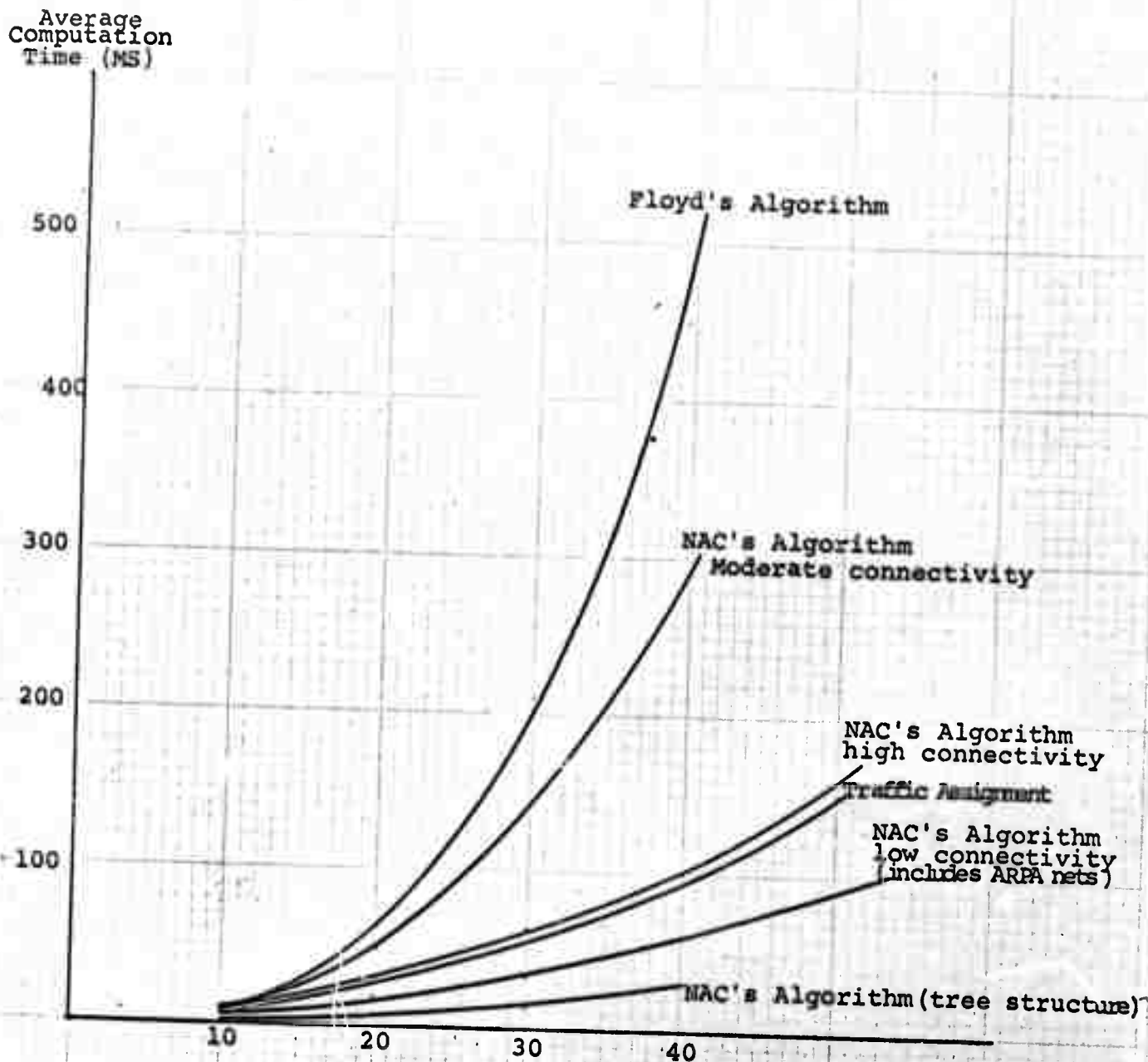


FIGURE 3.4.9

4.

RELIABILITY OF LARGE COMPUTER NETWORKS

4.1 INTRODUCTION

Spurred by the early successes of the ARPA Computer Network, and the economic potential of such nets, the size of computer networks can be expected to increase rapidly. Increased numbers of computers in a computer network give rise to several questions about its reliability. For the ARPA net the principal reliability requirement for smaller network designs (of less than 30 or 40 nodes) is that there exist two node disjoint paths between every pair of nodes in the net. This guarantees that at least two nodes or links must fail before any two nodes cannot communicate with one another. More detailed reliability analysis of networks designed in this way indicated that this approach does in fact guarantee sufficient reliability for the network taken as a whole for nets similar to the current ARPA net. The first question of interest is: does the two node disjoint path method suffice as the number of nodes grows. Closely related to this question is: what minimum amount of network investment is required as the number of nodes increases in order to maintain a given level of network reliability. In the next section results bearing on these questions are given. Simply stated, it does not appear that network reliability constraints can be met for

very large nets simply by requiring two node disjoint paths between each pair. More sophisticated techniques for designing large nets will be required. Such techniques are currently under investigation.

For many reasons it seems imperative that large computer nets will exhibit some hierarchical or decomposed structure. In Section 4.3 the computational consequences for reliability analysis of a two level hierarchal approach is investigated in which the nodes are partitioned into subnetworks called "regions" interconnected by a "global" network. In the same section, the related question of what size regions yields minimum computation for analysis is explored.

Finally, to make reliability analysis of large networks feasible, computational improvements over the techniques employed for the smaller networks must be developed. A detailed analysis of the growth of computation with network size for various reliability analysis techniques is found in Section 4.4, and new techniques which are much faster than previously known techniques are specified.

4.2. NETWORK RELIABILITY AS A FUNCTION OF SIZE

Low cost computer network designs for the ARPA Network have up to now been made under the reliability constraint that two node disjoint paths exist between every pair of nodes. For networks up to 30 or 40 nodes this criterion yields networks which are very

resistant to failures of their components. For example, for the 23 node 28 link ARPA Net, over 95% of all node pairs are able to communicate, on the average, if the nodes and links are down 2% of the time.

To measure how adequately this design technique maintained network reliability as the number of nodes in the network increased, low cost networks with 20, 40, 60, 80, 100 and 200 nodes respectively were designed using NAC's network design program with the reliability constraint of two node disjoint paths. The results are shown in Figure 4.2.1.

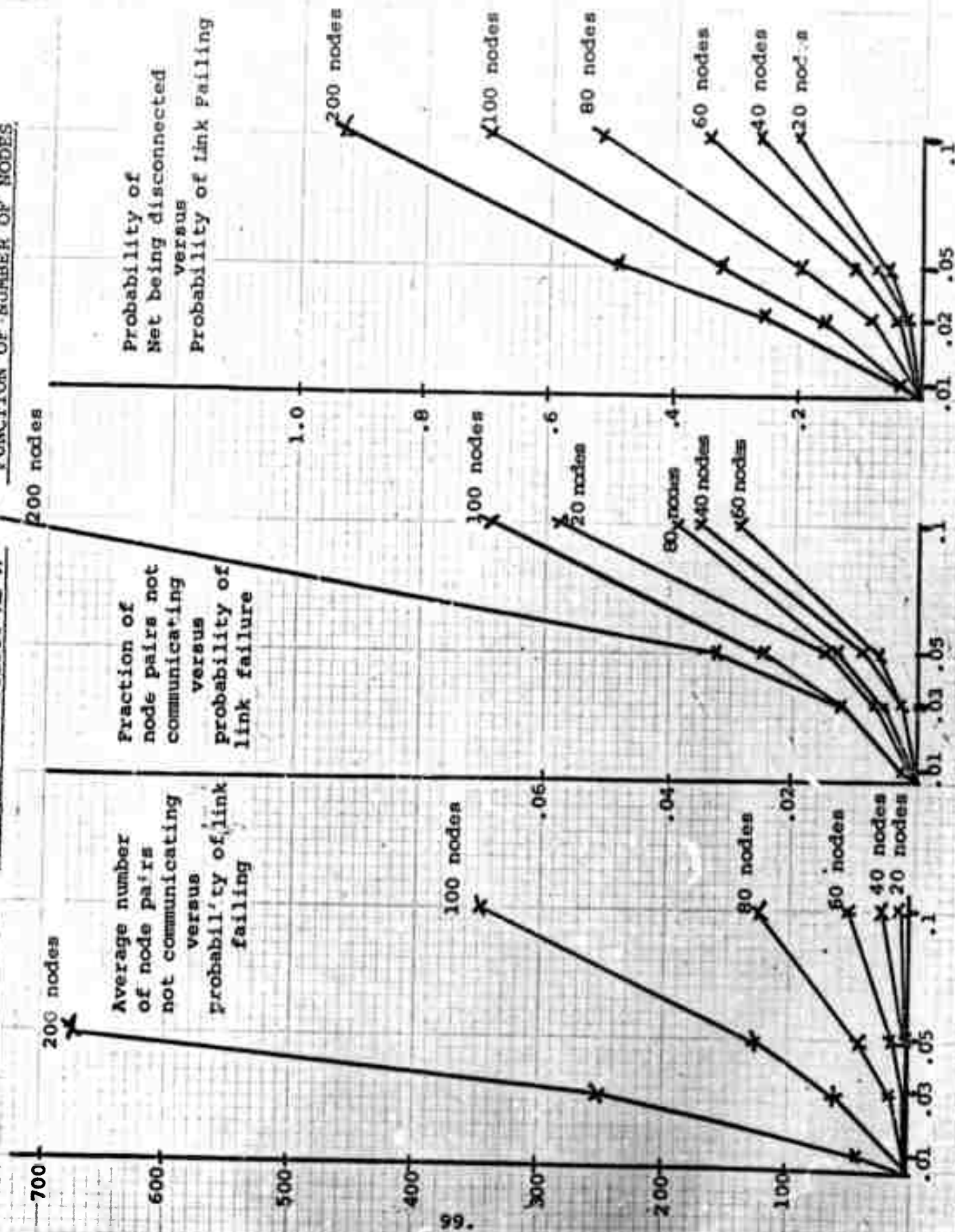
As measured by the fraction of node pairs not communicating, the reliability actually increased with the number of nodes up to 60 nodes at which point the reliability decreased with the number of nodes. Thus, at least for this series of networks, reliability will be degraded as the number of nodes increases above 60.

Another indication of the dependence of network reliability on the number of nodes is a result of Gelmans [1967]. If we consider nets in which only the links fail and take as our measure of reliability the probability that the net will be connected, Gelmans found that the number of links must increase at least as fast as $\frac{NN \ln NN}{2 |\ln p|}$ where NN is the number of nodes and p is the probability of link failure in order that the probability of network failure not approach 1. Since a 2 connected net has on the order of NN links

FIGURE 2.2.1

NETWORK RELIABILITY AS A

FUNCTION OF NUMBER OF NODES



the behavior of the term $(\ln NN)/2|\ln p|$ implies that as NN passes the point where $NN = 1/p^2$, 2 connectivity will no longer guarantee "sufficient" reliability. For example, if both nodes and links fail with probability 0.02, an approximate analysis with only link failures would indicate 2 connectivity becomes inadequate for NN approaching 100.

These two results while not definitive indicate strongly that more sophisticated methods of maintaining reliability for the designs of a larger networks must be developed.

4.3 COMPUTATIONAL CONSEQUENCES OF NETWORK DECOMPOSITION

We consider a two level hierarchical decomposition of networks. The nodes are divided into groups called regions. Nodes which are end nodes of links connecting two regions are called central nodes. A regional subnetwork or a local network is the nodes of a region together with the links which connect two nodes in the region. Such a decomposition may result from the necessity of reducing the analysis or design procedures to manageable size or it may be inherent in the structure of the network itself. For example, the routing procedure may require tables which are limited in size or the load on the network may be such that traffic within certain groups of nodes is much heavier than traffic between the groups; this leads naturally in both cases to hierarchical networks.

One of the first questions that arises is how large to make the regions and how many regions to create. One approach is to attempt to minimize the computational requirements for the design procedure. Suppose we have a procedure to perform on a network for which the computing time can be characterized as a polynomial function of the number of nodes. Two examples are the NAC network design algorithms and some of the NAC reliability analysis programs.

Suppose further that the procedure can be carried out by performing the procedure separately on each region of our decomposed network and then performing the procedure once on the global network. Let NN be the total number of nodes, NR the number of regions and NG the number of central nodes per region in the global network. We assume that the number of nodes is the same for each region as is the number of central nodes and that the computing time for a subnetwork with n nodes is proportional to n^v for some $v > 1$. While these assumptions are not usually exactly valid, the conclusions based on them are indicative and the techniques used can also be applied to more general cases.

Under these assumptions the total computation time is proportional to

$$T = NR \left(\frac{NN}{NR} \right)^v + (NR \cdot NG)^v,$$

where the first term corresponds to the regional calculation and the last term to the global calculation. Assuming that NR and

(NN/NR), the number of nodes per region, can take on non-integer values we can determine the number of regions which corresponds to minimum computer time by setting $\frac{\partial T}{\partial NR}$ to zero.

$$\frac{\partial T}{\partial (NR)} = NN^v (1-v) (NR)^{-v} + v (NR)^{v-1} NG^v = 0$$

implies

$$NR = \left(\frac{NN}{NG}\right)^{\frac{v}{2v-1}} \left(\frac{v-1}{v}\right)^{\frac{1}{2v-1}}$$

The asymptotic result as $v \rightarrow \infty$ is of interest

$$\lim_{v \rightarrow \infty} \left(\frac{NN}{NG}\right)^{\frac{v}{2v-1}} \left(\frac{v-1}{v}\right)^{\frac{1}{2v-1}} = \sqrt{\frac{NN}{NG}}$$

This result can also be obtained by noticing that for v large T is dominated by the subnetwork with the largest number of nodes; it is then easy to convince oneself that the minimum computation will occur when the global network has exactly as many nodes as the regions; i.e., when $NR \cdot NG = NN/NR$. Solving this for NR yields $NR = \sqrt{NN/NG}$. In Table 4.3.1 the optimal number of regions is given to the nearest integer for $v = 2, 3, 4, 5, 6, \infty$ and for $NN = 20, 40, 60, 80, 100, 200, 400, 600, 800, 1000$. NG was taken to be 2 in all cases.

Reliability analysis of decomposed networks can be carried out in a straightforward way. The basic step in reliability

TABLE 4.3.1

OPTIMUM NUMBER OF REGIONS FOR DECOMPOSITION

Order of Computation	Number of Nodes									
	<u>20</u>	<u>40</u>	<u>60</u>	<u>80</u>	<u>100</u>	<u>200</u>	<u>400</u>	<u>600</u>	<u>800</u>	<u>1000</u>
n^2	4	6	8	9	11	17	27	36	43	50
n^3	4	6	7	8	10	15	22	28	34	38
n^4	4	5	7	8	9	13	20	25	29	33
n^5	4	5	6	8	9	13	19	23	27	31
n^6	3	5	6	7	9	12	18	22	26	29
n^∞	3	4	5	6	7	10	14	17	20	22

analysis is the determination of the number and size of components in a graph. First, let us review the technique for networks without decomposition (see the Third Semiannual Report).

Algorithm 1.

- Step 0: (Initialization). Start with $A_0 = \emptyset$ and associate with node i a component label i . Set the number of elements in each of the NN components to 1. Set $k = 0$ and go to Step 1.
- Step 1: (New Link). Add a link $a_k = (m_k, n_k)$ to A_k to form A_{k+1} . If there are no remaining links; i.e., $A_k = A$, stop. Otherwise examine the labels of m_k and n_k . If they are the same, repeat Step 1 with $k := k+1$. If not, go to Step 2.
- Step 2: (Join Components). Change all the node labels which are the same as the label of m_k (including m_k 's label) to the label of n_k . Set the number of elements in the component corresponding to n_k to the sum of the component sizes previously corresponding to m_k and n_k . Set $k := k+1$; go to Step 1.

The computation for this algorithm in the naive form given here is dominated by the relabeling in Step 2 which in total involves on the order of NN^2 operations. (A detailed analysis and more efficient variants of this algorithm are discussed in the next section).

To use this algorithm on a decomposed network the algorithm is first applied to each region separately. Then the algorithm is applied to the global network with the following simple modifications to the Initialization Step 0. Each global node is given an initial node label which is a pair consisting of its region number and the component number it ended up with in the regional analysis. The number of elements in the component pairs is the number of elements in the components resulting from the regional analysis. With this initialization the algorithm proceeds as before.

For example, if $NG=2$, $NN=200$, the minimum computation (see Table 4.3.1) would occur when the number of regions is 17.

To test out the decomposition approach to network reliability analysis a new computer program was written. By making extensive use of list data structures decomposed networks with arbitrary numbers of regions and arbitrary size of global networks can be handled. The examples described in Section 2 were run using the program.

4.4 FAST ALGORITHMS FOR COMPUTING COMPONENTS OF SPARSE NETWORKS

In our previous work on reliability analysis of networks, three approaches were introduced. In this section we examine in some detail relative computational merits of these methods. The basic

problem is: given a network with NN nodes and NB links where each link has probability p of failing, we wish to estimate $h(p)$, the probability of the network being disconnected by the failing links, or to estimate $n(p)$, the expected number of node pairs not able to communicate, for NP values of p . The first approach which we shall call the naive approach is to first generate a random number for each link. If the number is less than p , the corresponding link is considered failed and removed from the net; otherwise the link is left in. Then the resulting network is checked for connectivity or the number of node pairs disconnected depending on whether $h(p)$ or $n(p)$ is desired. This computation is repeated enough times until a sufficiently accurate estimate is obtained according to the usual standards of Monte Carlo simulation. This entire procedure must then be repeated for each of the NP values of p of interest.

The second approach which we will call the functional simulation approach is based on a technique used for simulation approaches to percolation problems. Here as in the naive method, a random number is generated for each link. Now however, we order these numbers and their corresponding links in decreasing order. If r_i is the random number generated for the i th link suppose

$$r_{i_1}, r_{i_2}, \dots, r_{i_{NB}}$$

are the random numbers in decreasing order. Then in the naive

method if $r_{i_k} > p > r_{i_{k+1}}$, we would analyze for connectivity the subnetwork consisting of the links i_1, i_2, \dots, i_k . We can evaluate one point in the sample for each value of p for one set of random numbers using the following procedure: for $1 \geq p > r_{i_1}$ the network has no links for $r_{i_1} \geq p > r_{i_2}$ the network consists of the link i_1 only for $r_{i_2} \geq p > r_{i_3}$ the network consists of links i_2 and i_3 etc. What makes this technique especially effective is that there exist connectivity algorithms which analyze the network one link at a time with the links in arbitrary order. Thus, if the network with all the links present is analyzed for connectivity, introducing the links in the order i_1, i_2, \dots, i_{NB} using one of these techniques, we achieve the analysis for the subnetworks consisting of i_1, i_2, \dots, i_k for $k = 1, \dots, NB$. Thus in particular we can determine $h(p)$ or $n(p)$ for all NP values of p at once using a connectivity algorithm once for each sample point rather than NP times as would be required in the naive method. If only $h(p)$ is required, the computation simplifies even further, then for a given sample point the only quantity of interest is the smallest k for which i_1, i_2, \dots, i_k is connected. Then, for all $p > r_{i_k}$ the net is disconnected and for all $p \leq r_{i_k}$ the net is connected.

By a theorem due to Kruskal [1956] each connectivity calculation is equivalent to determining a maximum total length spanning forest where the length of a link i is r_i . This relation is discussed in detail in a forthcoming paper by Kershbaum and Van Slyke. [1972].

The final technique which we call the Moore-Shannon approach is based on the use of the equation

$$(1) \quad h(p) = \sum_{k=0}^{NB} C(k) p^{NB-k} q^k$$

where $C(k)$ is the number of disconnected subnetworks with exactly k links. This relation is similar to one due to Moore and Shannon [1956] for analyzing the reliability of switching networks. A similar relation can be used for $n(p)$:

$$(5) \quad n(p) = \sum_{k=0}^{NB} D(k) p^{NB-k} q^k$$

where $D(k)$ is the average number of node pairs not communicating over all subnetworks with exactly k links. In this approach each $C(k)$ or $D(k)$ is estimated by simulation rather than attempting to directly estimate the sums $h(p)$ or $n(p)$. As was shown in the Third Semi-annual Report, this is especially effective in (1) because all but $NB-NN+2$ of the $C(k)$ are known a priori. There are $\binom{NB}{k}$ subnetworks with exactly k links. If this number is reasonably small, $C(k)$ and $D(k)$ can be calculated by enumeration, and if the number is too large they can be estimated by simulation. In either case it is useful to use a connectivity algorithm which, after the connectivity of one subnetwork is determined, subnetworks similar

to the first can be easily analyzed using the results of the first analysis. Such an algorithm was described in the Third Report.

We now introduce several useful connectivity algorithms and analyze the computation required for each one. Before we begin, we note that for a network which is nearly complete, any connectivity algorithm will require at least a number of calculations on the order of NN^2 since each link must be examined and there are $(NN) \cdot (NN-1)/2$ possible links.

For the first algorithm we use the node adjacency representation of the network. Two nodes are adjacent if they are the endpoints of a link in the network. The node adjacency representation consists of specifying for each node, the nodes which are adjacent to it. There are two such entries for each link (i,j) one for i being adjacent to j and one for j being adjacent to i .

Algorithm 2:

Step 0: (Initialization) Set $i=1$, $j=1$, $k=1$, $S=\emptyset$. Label node $i=1$ with component label $j=1$. Go to Step 1.

Step 1: (Look at new link). Find the next node i' which is adjacent to i ; if there are none, go to Step 3. If node i' is not already in a component, go to Step 2. If the node i' is already labeled with a component number, repeat Step 1.

- Step 2: (Add a node to the current component) Label the node i' with the current component label j and add the index of the labeled node to the stack S . Return to Step 1.
- Step 3: (Scan a new node). Remove a node index i'' from S and set i equal to i'' . Go to Step 1. If S is empty, go to Step 4.
- Step 4: (Current component complete--start a new one) Set $k:=k+1$ if $k > NN$ we are done; otherwise, if node k is unlabeled, set i equal to k and set $j:=j+1$. Go to Step 1. If node k is labeled, repeat Step 4.

Algorithm 2 is close to being the most efficient algorithm possible for determining the components of a graph. It has the disadvantage that the links cannot be introduced in an arbitrary order. Thus this method is only appropriate for the naive approach to reliability analysis. To estimate the order of the computation involved we note that Step 1 is performed exactly $2 \cdot NB$ times (since each link appears twice in the node adjacency representation). Step 4 is repeated exactly NN times and Steps 2 and 3 are performed $NN-NC$ times where NC is the number of components. Thus, the number of computations is the sum of a term linear in NB and a term linear in NN . Since to determine the components of a graph we must, in general, look at all the links and label all the nodes, no algorithm can be much faster for determining the components of a graph.

Next, we consider several variations of Algorithm 1 which was described in Section 4.3. As indicated in that section, the dominant term in the number of operations for Algorithm 1 arose from Step 2, the relabeling, and was on the order of NN^2 . Step 2 occurs when a link joining two, up to now, disjoint components is encountered. In Algorithm 1 the component labels of the first component were all changed to those of the second component. If instead the number of elements of each component are maintained and the labels on the smallest component are changed, the order of the number of operations is $NN \log_2 NN$ rather than NN^2 . To prove this, let $f(n)$ be the maximum number of node relabelings required for any net with n nodes using the modified version of Algorithm 1. Then we have the following recurrence relation for $f(n)$:

$$(2) \quad f(n) = \max_{1 \leq k \leq [n/2]} \{f(k) + f(n-k) + k\}$$

where k is integer and $[n/2]$ is the greatest integer less than or equal to $n/2$.

Theorem

(i) $f(n)$ monotone increasing

(ii) $f(2^l) = l2^{l-1}$

(iii) $\lim_{n \rightarrow \infty} \frac{f(n)}{(n/2) \log_2 n} \rightarrow 1$

Lemma

$f(x) = \frac{x}{2} \log_2 x$ for $x > 0$ satisfies

$$(3) \quad f(y) = \max_{1 \leq x \leq y/2} \{f(x) + f(y-x) + x\} \text{ for } y \geq 1.$$

Proof: Let $g(x; y) = (x/2) \log_2 x + ((y-x)/2) \log_2 (y-x) + x$.

$$\text{Then } \frac{d^2}{dx^2} g(x; y) = \frac{1}{2 \ln 2} \left\{ \frac{y}{x(y-x)} \right\} > 0 \text{ for } x < y.$$

Thus $g(x; y)$ is convex upward on $[1, y/2]$ and hence achieves its maximum in x at either 1 or $y/2$.

$$g(1; y) = ((y-1)/2) \log_2 (y-1) + 1 < (y/2) \log_2 y = g(y/2; y).$$

Thus the maximum is achieved at $x = y/2$ and

$$\max_x g(y; x) = f(y) = (y/2) \log_2 y.$$

Proof: (of Theorem) (i) is easily proven by reference to Equation (2) with $k = 1$. Thus,

$$f(n) \geq f(n-1) + f(1) + 1.$$

Next, we prove by induction on n that

$$f(n) \leq (n/2) \log_2 n$$

We have for $n=1$ $f(1) = (1/2)\log_2 1 = 0$. In general,

$$\begin{aligned} f(n) &= \max_{1 \leq k \leq \lfloor n/2 \rfloor} \{ f(k) + f(n-k) + k \} \\ &\leq \max_{1 \leq k \leq \lfloor n/2 \rfloor} \left\{ (k/2) \log_2 k + ((n-k)/2) \log_2 (n-k) + k \right\} \\ &\leq \max_{1 \leq x \leq \lfloor n/2 \rfloor} \left\{ (x/2) \log_2 x + ((n-x)/2) \log_2 (n-x) + x \right\} \\ &= (n/2) \log_2 n, \end{aligned}$$

where the first inequality results from the induction hypothesis, the second is a consequence of simply allowing more points to be considered in the maximization, and the last equality results from the lemma. Now to prove (ii), we do this by induction on ℓ . For $\ell=1$ we have the trivial result $f(2) = 1 \cdot 2^0 = 1$ since there are only two nets on 2 nodes. In general, by Equation (3) for $n = 2^\ell$,

$$\begin{aligned} \ell \cdot 2^{\ell-1} &\geq f(2^\ell) \geq \max_{1 \leq k \leq 2^{\ell-1}} \{ f(k) + f(n-k) + k \} \\ &= f(2^{\ell-1}) + f(2^{\ell-1}) + 2^{\ell-1} \\ &= (\ell-1)2^{\ell-2} + (\ell-1)2^{\ell-2} + 2^{\ell-1} \\ &= \ell 2^{\ell-1} \text{ by the induction hypothesis.} \end{aligned}$$

This establishes (ii). (iii) is an obvious consequence of (i) and (ii).

The number of operations can be reduced still further using another variant, Algorithm 3, of Algorithm 1. The key to

Algorithm 3 is the use of a tree structure to maintain the number of the components of each node. Associated with each node i is another node, $f(i)$, the father of i . To find the component of a node i , set $i_0 = i$ and repeat the iteration:

	$i_{k+1} = f(i_k)$	(*)
until	$i^* = f(i_k) = i_k,$	(**)

Then, i^* is the component label. Define $F(i) = i^*$ to be the patriarch of i . The sequence i, i_1, \dots, i_k, i^* will be called the chain determined by i . Then there is a one to one correspondence between components and patriarchs. Based on such a data structure, a simple algorithm for determining the components might be:

Algorithm [Read: 1969]

Step 0: (Initialization). Set $f(i) := i, i=1, \dots, NN, A_0 = \emptyset, k:=0$ and go to Step 1.

Step 1: (New link). Add a link $a_k = (m_k, n_k)$ to A_k . Form A_{k+1} . If there are no remaining links, i.e., $A_k = A$, Stop. Examine $F(m_k)$ and $F(n_k)$. If they are identical, repeat Step 1 with $k := k+1$. If not go to Step 2.

Step 2: (Join trees). Set $f(F(n_k))$ to $F(m_k)$. Set $k:=k+1$. go to Step 1.

Analysis of Algorithm: The major computational expense occurs in Step 1. Step 1 is encountered NB times and each time $F(n_k)$ and $F(m_k)$ must be evaluated. This could involve examining at most $2k$ nodes if n_k and m_k are in the same component and k nodes if they are in different components. For example, if the net is a chain and the links are introduced in order then k nodes would be examined for each k . In any case, the order of the calculation is $NB \cdot NN$ so this algorithm is inferior to Algorithm 2. However, we can improve this algorithm so that it takes a number of operations proportional to NB by using the information gained in evaluating F to collapse the tree.

Algorithm 3:

- Step 0: (Initialization). Set $f(i) := i$ for $i = 1, \dots, NN$,
 $A := \emptyset$, $k := 0$ and go to Step 1.
- Step 1: (New link). Add a link $a_k = (m_k, n_k)$ to A_k to form A_{k+1} .
 If there are no remaining links, i.e., $A_k = A$, go to Step 2. Set $f(i) := F(m_k)$ for every node on the chains defined by m_k and n_k . Set $k := k+1$ and do Step 1 again.
- Step 2: (Clean up). Set $f(i) := F(i)$ for $i = 1, \dots, NN$

Analysis of Algorithm 3:

The detailed analysis is somewhat tedious and can be found in [Kershenbaum & Van Slyke: 1972]. The principal computational expense

is in collapsing the tree and computing $F(m_k)$. In particular, we must essentially travel the chain from m_k twice or save the chain because we cannot relabel the chain until we know $F(m_k)$. Nevertheless, the total number of operations can be shown to be of the order of NB . To obtain intermediate results on the components one keeps the current number of components and associates with each patriarch node p a number $d(p)$ which represents the number of nodes "down" the component tree from p where $f(i)$ is "above" i . Then in Step 1 if $F(m_k) \neq F(n_k)$ then two trees are being joined so the number of components is reduced by one and $d(F(m_k)) := d(F(m_k)) + d(F(n_k))$. Then for any patriarch p the number of elements in the corresponding component is $d(p)$.

The father relations established by Read's algorithm can be thought of as branches of a tree, T , where if $j = f(i)$, then j is "above" or is the "predecessor" of i in T . Initially, no branches of T are present. As each step is performed a branch of T is "filled in" if 2 components are connected, otherwise nothing happens. In the sample tree T depicted in Figure 4.4.1 the heavy lines represent the branches "filled in" so far at some stage of the algorithm. A step in which the link $(5, 8)$ is considered would cause $(2, 7)$ to be filled in, while the introduction of $(5, 6)$ would leave the situation unchanged. In Algorithm 3 we initially

consider the same T as in Read's Algorithm. However, the structure of T may be altered as a consequence of any step of Algorithm 3. Thus, suppose the tree T depicted in Figure 4.4.1 represents the component tree at some point during the application of Algorithm 3. Then a step in which $(5,8)$ is introduced would result in the modified tree T' of Figure 4.4.2. All nodes whose fathers are examined in the course of determining $F(m_k)$ and $F(n_k)$ become immediate successors of $F(m_k)$. In the example, these are nodes 5, 4, 2, 8, 7.

Algorithm 3 while being somewhat more complicated than Algorithm 1 has the virtue that links can be introduced in arbitrary order and at any point in the algorithm the subnet is completely analyzed. In particular, for the functional simulation method the links can be introduced in order of the random numbers assigned to the links. Moreover, the order of the number of operations is still NB.

A final algorithm we consider is Algorithm 4, Prim's Algorithm. [Prim:1957], [Dijkstra: 1959]. It is designed to find a maximum length spanning tree. We modify this algorithm for use in reliability analysis. At step k we have a set of nodes A_k and we consider the links with one incident node in A_k and the other one not in A_k . If, of all such links, (i,j) is the longest with $i \in A_k$, $j \notin A_k$ we then let $A_{k+1} = \{j\} \cup A_k$ and iterate. Stated more formally:

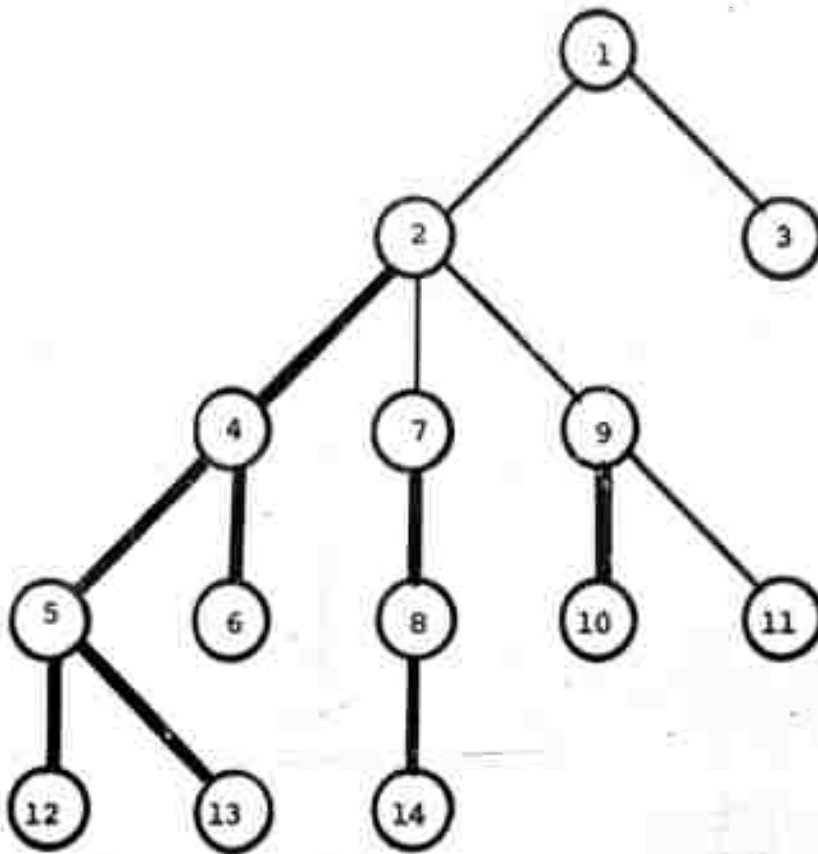


FIGURE 4.4.1

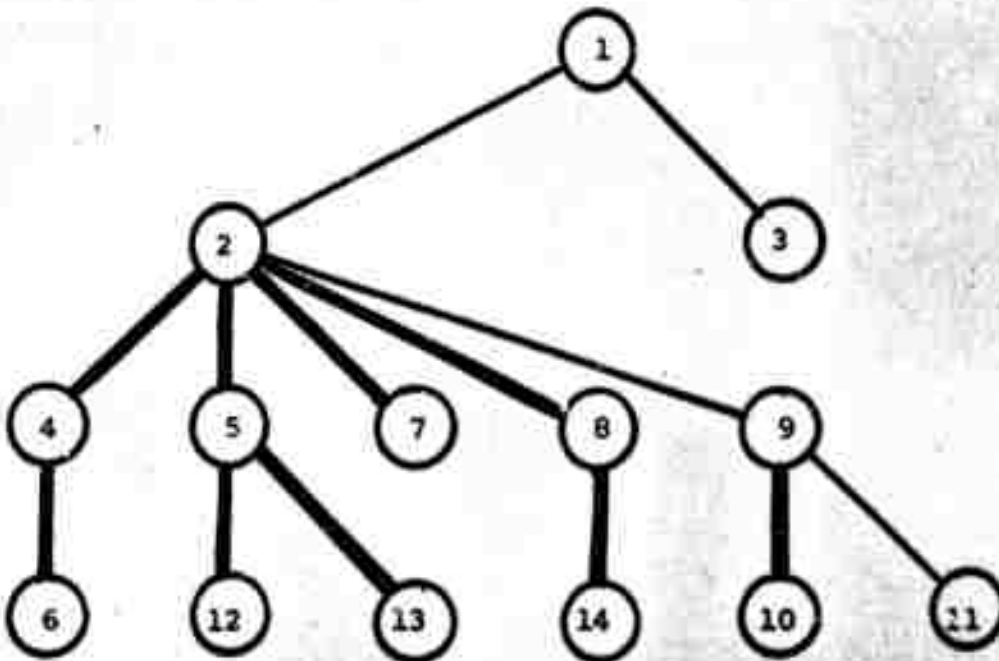


FIGURE 4.4.2

Algorithm 4: Let d_j be the length of the longest link from A_k to j for $j \notin A_k$.

Step 0: (Initialization). $A_1 = \{1\}$, $k = 1$, $d_1 = 0$, $d_j = r_{1,j}$ if $(1,j)$ is a link, $d_j = \infty$ otherwise, where $r_{i,j}$ is the random number assigned to link (i,j) .

Step 1: (Increase A_k). Let $d_{j^*} = \max_{i \in S_k} d_i$ and $A_{k+1} = A_k \cup \{j^*\}$.

Update the d_j by $d_j := \max \{d_j, r_{j,j^*}\}$. Go to Step 2.

Step 2: (Test for termination). If $k = NN$, stop; otherwise go to Step 1.

Algorithm 4 can be used if the measure of network reliability is the probability of the network being connected. If one saves the smallest d_{j^*} encountered in Step 1, (suppose it is r_k^*) then for $p < r^*$ the net is connected for that sample point and for $p \geq r^*$ the net is disconnected. This algorithm restricts the order in which the links can be introduced but does not require one to present the links in order of decreasing $r_{i,j}$. Step 1 is performed $NN-1$ times and at the k th step $2 \cdot NN - 2k - 1$ comparisons must be made to determine j^* and to update the d_j . Thus about $NN(NN-1)$ comparisons must be made which gives an order of computation of NN^2 for this algorithm.

Evaluation of Algorithms for Reliability Computations

We are now in a position to estimate the most efficient techniques for large network reliability problems. In order to find NP values of $h(p)$ and $n(p)$ for a network with NN nodes and NB links, we require the following amounts of computation. For the naive method, we perform on the order of $NP \cdot NB$ operations for each sample point. To use functional simulation, we must presort the links by decreasing values of their random numbers. This computation, which in itself requires on the order of $NB \log_2 NB$ comparisons dominates the order of computation for Algorithm 3 which is essentially linear in NB. If reliability is measured by the probability of disconnection, Algorithm 4 can be used to avoid the ordering. The order of calculations in this case is NN^2 .

The Moore-Shannon approach cannot be directly compared because the statistical analysis is different. However, Algorithm 1 or Algorithm 3 could be used effectively by generating a random ordering of the links and then introducing them one at a time to obtain a sample point for $C(1)$, then $C(2)$, ..., etc. This is another means of avoiding the sorting of random numbers associated with links.

Limiting ourselves to the naive method and the functional simulation method, we see that for NP small, naive simulation is preferable because it avoids sorting the links. For NP large function simulation is preferable using Algorithm 3 except that

for nearly complete networks using the connectivity criterion Prim's Algorithm should be used since for complete networks the $NB \log_2 NB$ for sorting will eventually overtake the NN^2 operations required for Prim's algorithm.

The exact points at which these tradeoffs occur depend on characteristics of the computer and the coding used to implement the procedures. Some empirical studies are presently in progress and the results will be reported at a later date.

REFERENCES

- 1) W. Chou and H. Frank, "Survivable Communication Networks and the Terminal Capacity Matrix," IEEE Trans. on Circuit Theory, Vol. CT-17, No. 2, May 1970.
- 2) E. W. Dijkstra, "A Note on Two Problems in Connection with Graphs," Numerische Mathematik, 1, pp. 269-271, 1959.
- 3) R. W. Floyd, "Algorithm 97, Shortest Paths," Communication of ACM 5, 345, 1962.
- 4) T. C. Hu, "Revised Matrix Algorithm for Shortest Paths in a Network," SIAM J. 207-218, January 1967.
- 5) A. Kershenbaum and R. Van Slyke, "Minimum Spanning Forests in Sparse Graphs," in preparation, 1972.
- 6) J. B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," Proceedings of the American Mathematical Society, 7, pp. 48-50, 1956.
- 7) Network Analysis Corporation, "Research in Store-and-Forward Computer Networks," Semiannual Report No. 1, June 1970 (Contract No. DAHCl5-70-C-0120)
- 8) Network Analysis Corporation, "Research in Store-and-Forward Computer Networks," Semiannual Report No. 2, Dec. 1970, (Contract No. DAHCl5-70-C-0120)
- 9) Network Analysis Corporation, "Research in Store-and-Forward Computer Networks," Semiannual Report No. 3, June 1971 (Contract No. DAHCl5-70-C-0120)
- 10) R. C. Prim, "Shortest Connection Networks and Some Generalizations," Bell System Tech. J., pp. 1389-1401, Nov. 1957.

END